1997

A Micro-Mechanical Fatigue Damage Model for Uni-Directional Metal Matrix Composites.

Rainer Echle
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

A MICRO-MECHANICAL FATIGUE DAMAGE MODEL FOR UNI-DIRECTIONAL METAL MATRIX COMPOSITES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Department of Civil and Environmental Engineering

by
Rainer Echle
Dipl.Ing. (FH), Fachhochschule Karlsruhe, Germany, 1986
M.S., Louisiana State University, May 1990
May 1997

Acknowledgements

I would like to thank Boyd Professor George Z. Voyiadjis for his continuous support and guidance in conducting this research. His suggestions and discussions throughout the course of this research have been invaluable. As an advisor and mentor he provided the necessary impetus and direction for many thoughts and ideas. My sincere thanks go to him.

I also would like to thank Dr. Robert Dorroh, Department of Mathematics, Dr. R. Richard Avent, Chairman of the Department of Civil and Environmental Engineering, Dr. Su-Seng Pang, Department of Mechanical Engineering, and Dr. Jianchao Jim Zhu, Department of Electrical Engineering, for taking their time and effort to serve on my dissertation committee. I would like to extend my thanks to Drs. Luis A. De Bejar and Patrick Zuraski, who have been serving on my general examination committee.

The support of the Department of Civil and Environmental Engineering for providing me financial support in the form of a teaching assistantship is greatly appreciated. I thank the staff in the Department of Civil and Environmental Engineering for their help and support, making my life easier at numerous occasions.

I would like to thank my colleagues Dr. Ganesh Thiagarajan, Dr. Taehyo Park, Dr. Anthony R. Venson and Dr. Vibhas Aravamuthan for the numerous fruitful discussions and helpful suggestions throughout the course of this work.

Special thanks go to my friends Dr. Anand Jagadeesh Puppala and his wife Padmaja, for their warm and special friendship over the past years until this day, and to Ganesh Thiagarajan and his wife Loganayake for their sincere friendship and

their support throughout the years to this day. I would also like to thank my dear friends Elias Khalaff, Elfriede Ott, Ghassan Alkadi, Vibhas Aravamuthan, Anthony Venson, Susheel Puppala, Anil Dedhia, Dr. James Bolner and his wife Smittie, who have made my time away from work so pleasant and whom I shared so many wonderful moments with. Their friendship will always be cherished and remembered. Thank you to all my friends over the years whom I did not mention here.

The special friendships to Carolina and John with their children Nathalie, Eileen and Veronica, to Luisa-Victoria and Mauricio with their daughter Isabella, have brought so many wonderful moments to my life, for which I cannot thank them enough.

My sincere thanks go to my beloved parents for their never ending love, care and support, for their sacrifices allowing me to pursue my education. Thank you so much to my brother and my sisters together with their partners and children, my close friends Jens and Silvia, for their continuous support and for always believing in me. The love and care of my family in law and my friends in Argentina with whom I shared so many wonderful moments, has given me so much throughout the course of this work. Thank you to all of you.

And finally, I extend my sincere thanks to my friend and partner for life, my wife Gabriela, for the patience and understanding she has shown throughout the course of this work, for keeping up with me despite the numerous hours, days, weeks and months I had to dedicate to my work rather than spending time together. Without her sincere love and care I would not have succeeded.

To Gugui,

To My Mom and Dad,

Rainer

Contents

List of Tables

List of Figures

Abstract

A micro-mechanical damage model for high cycle fatigue loading based on thermo-dynamical principles is developed for uni-directional continuous fiber reinforced metal matrix composites. The model uses a micro-mechanical based theory to predict the fatigue damage evolution in the individual constituents of the composite material, namely the fibers and the matrix, from which the overall fatigue damage evolution in the composite is obtained. The micro-mechanical analysis is performed for each individual constituent using stress and strain concentration tensors based on the Mori-Tanaka method. A fatigue damage criterion based on thermo-dynamical principles is developed and applied to each of the constituents. Fatigue damage evolution equations are derived for the individual constituents and appropriate damage model parameters are established which reflect the physical behavior of the constituents with respect to damage evolution during the fatigue life of the material. The developed model is implemented into a numerical simulation code which is then used to simulate several fatigue tests for a uni-directional metal matrix composite system. High cycle fatigue loading is only modeled here, which is characterized by elastic deformations at the macroscale. The fatigue loading is applied as a uni-axial normal stress in the fiber direction in the form of a sinusoidal wave. A parametric study is conducted to show the influence of the model parameters on the damage evolution process in the constituents. These parametric studies show that the model is able to capture different dominating failure modes in the composite material. Finally, several complete fatigue simulations are performed. Results from these simulations are presented in the form of damage evolution curves for the individual constituents

as well as the overall composite material. The obtained results show qualitative good behavior with respect to the physical behavior observed in fatigue experiments. The failure mode for the investigated composite material system is captured and displayed properly. Furthermore, a comparison with available experimental data is made for the number of cycles to failure obtained from the simulations. This comparison is shown in the form of a *Wöhler diagram* and satisfactory agreement is observed.

Chapter 1
Introduction

With the increase in performance of aerospace vehicles, design factors such as weight and material strength play an increased role in the design philosophies of such vehicles in order to increase fuel efficiency as well as speed. Along with such drastic performance enhancements appropriate materials need to be investigated and developed that are capable of sustaining the arising loads and of performing safely under conditions such as those occurring during flight, while retaining their structural integrity. Candidate materials for such advanced applications have been identified among the composite materials, especially in the area of Metal Matrix Composites (MMC's).

Among the MMC's special consideration has been given to continuous fiber reinforced Titanium Matrix Composites (TMC) especially due to the fact that these materials maintain their excellent strength to density ratio even at elevated temperatures. This intrinsic material property has drawn wide attention to TMC's especially from the gas turbine engine manufacturing industry for potential use of such advanced materials in a new generation of gas turbine engines. The main reason for this success is attributed to the tremendous reduction in weight of key engine components resulting in an increase in engine performance as well as in a reduced fuel consumption. Titanium matrix composites offer higher mechanical properties, better dimensional stability, and strength retention even at elevated temperatures, such as those occurring in gas turbine engines, as compared to their monolithic counterparts.

Nevertheless, the general use and employment of MMC's and TMC's still has major drawbacks. First, the production/manufacturing costs for such materials are still high due to the special manufacturing processes involved. Second the employment of such materials in vital components of an aircraft or space vehicle, such as a turbine engine, requires a thorough understanding and control of the material behavior under extreme loading conditions such as those occurring during the regular service life of such vehicles. This calls for the development of material models which are capable of predicting real life behavior of such materials with a deterministic margin of risk. As of today the behavior of such MMC's and TMC's even under normal loading conditions and temperature environments is not yet fully understood and appropriate material models still lack reliability and general applicability as compared to those of their monolithic counterparts.

Through the last two decades considerable experimental as well as theoretical research effort has been put forward to better understand and analytically model metal matrix composites and hence titanium matrix composites in order to allow for a more efficient use of the tremendous potential contained in these advanced materials. Specific focus has been directed at the modeling of the real material behavior using the concept of continuum damage mechanics (CDM) originally introduced by Kachanov (1958) for the creep behavior of isotropic materials. The concept of continuum damage mechanics describes the phenomenon of progressive material degradation during the loading process due to damage development resulting ultimately in rupture upon continuation of increasing the applied loads.

Damage itself has been defined and described physically by means of voids, cracks, cavities, etc., leading to a reduction of load resisting area/volume and hence enhancement of internal stresses denoted effective stresses. Damage is not directly accessible

for measurement. However, it may be evaluated quantitatively by observing and measuring the change in density (Jonas and Baudelet, 1977), the change in resistivity (Cailletaud et al., 1980), the reduction in strength and elastic stiffness (Johnson, 1983; Charewicz and Daniel, 1985; Lemaitre, 1985a; Wu et al., 1996; Wang et al., 1996), the reduction in toughness, stability and residual life (Marco and Starkey, 1954; Woodford, 1973) or reduction in the fatigue limit (Henry, 1955; Gatts, 1961; Bui-Quoc et al., 1971) of the material as loading progresses. The major problem in modeling material behavior which includes explicitly the effect of damage, is to find and define appropriate mechanical variables which enable the explanation of the material degradation from a mechanistic point of view. Furthermore such variables should allow to describe the material degradation within the framework of continuum damage mechanics. In the case of composite materials further complexity is added due to the consideration of the heterogeneous nature of such a material system itself, mainly the material phases consisting of matrix and fibers. Additional complexity is added by including the fiber-matrix interface as an additional material phase.

In principle there exist two different approaches to model the material behavior of composite systems. In the first approach the composite material is considered as a single entity. Hence, no distinction is made among the material phases. The material is described mechanically by averaging and hence smearing of the material properties of the constituents to arrive at the composite material properties. The averaging process is mainly done by employing certain homogenization procedures such as the simple rule of mixtures, the self-consistent method (Hill, 1965; Budiansky, 1965; Willis, 1977) or the Mori-Tanaka method (Benveniste, 1987; Norris, 1989; Lagoudas et al., 1991; Chen et al., 1992). The material behavior is then modeled for the entire composite system as a homogeneous material. However, the material

is modeled with averaged material properties obtained by means of the employed homogenization procedure. Material degradation in the form of damage is then considered and modeled as for a homogeneous material. Hence any damage occurring in the material during loading is treated as damage occurring in a homogeneous material and no distinction is made for different damage modes.

In the second approach the heterogeneity of the material is directly considered in the modeling of the composite material behavior. Through the application of stress and strain concentration tensors, which permit the distribution of external applied loads or displacements to the individual constituents, it is possible to model the material behavior of each constituent at an individual level before employing homogenization procedures to predict the overall composite behavior. Hence it is possible to model the material degradation of each constituent individually. The big advantage of such an approach is that it allows one to distinguish among the different damage modes and phenomena (matrix cracking, fiber cracking, debonding, delamination) as observed in experiments (Johnson, 1989; Jeng et al., 1991a,b; Neu, 1993). However, the complexity of the problem to be solved and the appropriate material model is drastically increased. Using then kinetic as well as kinematic boundary conditions the overall behavior and damage evolution of the composite material system maybe obtained based on the material behavior and damage evolution of the individual constituents, the fibers and the matrix, respectively. However problems arise when associating damage modes such as debonding with either constituent, namely the matrix or the fibers. Therefore it is necessary to consider whether it would be even more advantageous to introduce a third material phase in the form of an interface layer separating the matrix from the fiber. This new material phase could then be used to describe debonding through appropriate damage variables. It is obvious that

such an approach would further enhance the complexity of the model but would also allow to more realistically describe the composite material behavior.

As mentioned earlier, modeling the material behavior due to applied loading requires one to include the effects of material deterioration due to damage into the constitutive theory. In order to do so, it is necessary to define mechanical variables which describe the physical damage phenomena. The subject of defining and quantifying such damage variables itself has been the focus of vast research efforts in the past. Researchers proposed various damage variables such as the original scalar damage variable proposed by Kachanov (1958) for creep damage, second order damage tensors (Cordebois and Sidoroff, 1979; Sidoroff, 1980; Murakami and Ohno, 1980; Murakami, 1983)), fourth order damage tensors (Chaboche, 1978) and eighth order damage tensors (Chaboche, 1979). The definition of higher order damage tensors is necessary in order to be able to describe the damage phenomenon in its most general form, hence allowing for anisotropic damage development and evolution in materials. In the case of composite materials such a general form of a damage variable is vital in order to capture and model the material behavior more realistically. However, as usual, with the increasing complexity of the proposed damage variable rises the difficulty to relate these mechanically defined variables to physical quantities observed and quantified in experimental investigations. Efforts to quantify and relate physical defects in a material to a mechanical damage variable have been made for metals (Dyson and McLean, 1977; Levaillant and Pineau, 1982; Cailletaud and Levaillant, 1984; Hua and Socie, 1984) by quantifying inter-granular cavities and surface micro-cracks. In the case of metal matrix composites, Voyiadjis et al. (1993), Venson (1994), Voyiadjis and Venson (1995) and Voyiadjis et al. (1996) relate the physical damage in the form of crack densities to a mechanically defined damage variable in the form of a second order tensor.

Material degradation due to damage is closely related to the type of applied loading since the type of applied loading may add to and enhance the severity of certain damage modes. In engineering design fatigue loading of any kind is among the most severe loading cases for structural members (Dieter, 1988). Therefore it is not surprising that fatigue related problems and failures in engineering structures and components have been reported as early as 1866 by Wöhler (Wöhler, 1866). He reported the failure of railroad axles due to fatigue loading. Since then fatigue phenomena in engineering materials and the prediction of material failure due to fatigue loading have been the subject of enormous research efforts (refer to Chapter 2). With the advancement in technology and manufacturing, the severity of fatigue related problems in engineering has increased drastically due to the ever rising ambition of engineers to reduce and keep the use of materials in structural applications to the bare minimum necessary. The latter fact holds for both statically as well as dynamically loaded structures. With the advent of new and more advanced materials systems such as composites, the fatigue problem has reached new levels due to the more complicated material structure as compared to the conventional materials.

Numerous material models for fatigue loading have been developed for metals and alike materials as well as for composite materials. The majority of the models are phenomenological in nature and find their basis in observations made during experimental investigations and conclusions drawn from the analyzed experimental results. Especially for composite materials, where the micro structure of the material adds additional complexity, fatigue damage and life time prediction models have been developed for specific composite layups (Johnson and Wallis, 1984a,b; Johnson, 1993; Hart and Mall, 1995; Liaw et al., 1995; Chiang et al., 1995) mostly of special interest to the industry at the time of the investigation. Very few of the composite

fatigue damage models (see chapter 2) try to model and understand the fatigue phenomenon of uni-directionally reinforced composites. These are the base modules for any general composite layup. Among these the phenomenological approach is predominant. However, none of the models describes the fatigue phenomenon at the micro-structural (fiber - matrix) level.

The present research describes the development of a consistent true micro-mechanical fatigue damage model for uni-directional metal matrix composites based on thermo-dynamical principles. This model may be considered as a first step towards the development of a universal fatigue damage model for general metal matrix composites. The present model incorporates the influence of the micro-structure, fiber-matrix, of the composite system. Constitutive equations incorporating the effect of damage development and evolution are developed at the constituent level and then used to predict the overall behavior of the composite material system. Fatigue damage evolution equations are developed for each of the constituents in the material system to model damage development and evolution at the constituent level. The overall composite material behavior is then obtained by using a homogenization procedure, specifically the Mori-Tanaka scheme. Numerical results obtained from the implemented fatigue model include a continuous damage prediction in the individual constituents as well as in the overall composite over the entire fatigue life, non-existent at this time in other models. Furthermore the model allows for the simulation of different failure modes in a uni-directional metal matrix composite, such as matrix or fiber dominated failure. Comparison of fatigue life time of a few run-out simulations is provided to show the predictive capabilities of the model.

Following the outline in the introduction the organization of this document is as follows. Chapter 2 provides an introduction to the fatigue problem in engineering. The focus is then directed at the development of cumulative fatigue damage

models which have been mainly established for isotropic materials and then adapted for composite materials. A rationale is established for the motivation of the current research. Chapter 3 introduces and defines some basic characteristics and as well as some fundamental relations in the area of continuum damage mechanics. Furthermore the subject of micro-mechanical modeling is elaborated upon and used as the foundation for the micro-mechanical fatigue damage model developed and described subsequently. Chapter 4 is dedicated to the definition of material and model parameters. Especially the determination of the damage parameters as well as their influence on the damage evolution process is given special attention. Numerical results obtained from model analyses are described and discussed in chapter 5. In Chapter 6 the findings and conclusions of this work are summarized and appropriate provisions for future research in this area are made.

Chapter 2

Fatigue in the Literature

Reviewing the literature on the subject of fatigue in engineering materials, with emphasis on metal based materials, reveals that the explanation of fatigue phenomena and the prediction of fatigue life of engineering components and structures has been the focus of immense research efforts for more than a century. Fatigue phenomenon have been discussed as early as 1866 (Wöhler, 1866, 1870) who recognized the potential problems of repetitive loading due to the failure of railway axles. With the publication of his results (Wöhler, 1870) in form of charts he provided a design tool for fatigue problems which is still the basis for today's design charts known as the "Wöhler curves". His work was the initiation for many investigations in the field of fatigue problems. An excellent review on the general history of the fatigue problem in engineering is given by Schütz (1996).

Considering the fatigue problem in modern day science there are two major analytical approaches followed in order to explain and predict the phenomenon of fatigue in engineering materials, the phenomenological approach and the crack propagation approach, respectively. The former is concerned with lifetime prediction for complex loading histories using existing lifetime test data, mostly $S - N$ data obtained from experiments. The second approach is concerned with predicting the growth of a dominant crack of an initial predetermined size to a critical dimension due to the applied load (Suresh, 1991). This phenomenon, however, is only valid for metals and alike materials. For composite materials fatigue failure occurs due to various mechanisms

at the microscale, such as fiber cracking, matrix micro-cracking, fiber debonding and delamination (Chaboche, 1988a), which is mainly due to their heterogeneous nature. Difficulties in detecting and identifying such mechanisms in composite materials gave the prominence to the phenomenological approach when attempting to model fatigue lifetime for such materials.

Numerous linear and non-linear phenomenological models that are used to predict fatigue lifetime of metals subjected to constant as well as variable amplitude cyclic loading have been proposed based on the cumulative damage theory (Marco and Starkey, 1954; Henry, 1955; Corten and Dolan, 1956; Gatts, 1961; Marin, 1962). The cumulative damage theories attempt to assess the damage produced and its accumulation due to loading cycles of constant and varying stress amplitude. Miner (1945) introduced the cumulative damage theory for isotropic materials based on the earlier work by Palmgren (1924, 1945). In his work damage is considered as the reduction in fatigue lifetime due to the number of applied loading cycles. His cumulative damage theory was independent of the applied stress and the accumulation was linear. This simple linear damage rule also known as the *Palmgren-Miner*'s rule is given as

$$D = \frac{n}{N} \tag{2.1}$$

where the parameters n and N represent the number of actual load cycles and the number of cycles at the fatigue limit, respectively. It is obvious that in this case material failure due to fatigue is consistent with a damage value of $D = 1$. When expanding his work to include the case of multi-stress level fatigue loading Miner made the following assumptions:

1. Each stress level contributes an amount of damage given by the linear cycle

ratio of the stress level,

2. Damage caused by each stress level is independent of any previous stress history, and

3. The total damage due to the total loading is equal to the sum of the damages due to the individual stress levels.

Using these assumptions, also known as the equivalent cycle hypothesis, the remaining fatigue life of a material subjected to a multi-stress level fatigue loading may be predicted. For the case of a two-stress level fatigue loading the remaining fatigue life n_r is obtained as

$$n_r = N_2 \left(1 - \frac{n_1}{N_1}\right) \tag{2.2}$$

and for the general case of a multi-stress level fatigue loading of n levels the remaining fatigue life is simply given as

$$n_r = N_m \left(1 - \sum_{i=1}^{m-1} \frac{n_i}{N_i}\right) \tag{2.3}$$

The simplicity involved in this rule makes it still a favored tool for fatigue life-time prediction even in current design practices. However it should be emphasized that a major drawback of this rule is that it fails to predict the effect of load history. Experimental data (Hwang and Han, 1986b) showed that the order in which stress levels are applied, does have significant influence on the fatigue behavior and fatigue lifetime of materials.

Based on the concept of damage accumulation numerous non-linear cumulative fatigue models have been proposed by Marco and Starkey (1954), Henry (1955), Gatts (1961), Chaboche (1974), Manson (1979), Lesne and Savalle (1987) and Chaboche

and Lesne (1988). Various philosophies were followed when developing such models. Miller and Zachariah (1977) considered the separation of the fatigue life into two distinct phases (crack initiation and crack propagation), Bui-Quoc et al. (1971) based their model on the decrease in the fatigue limit and Chaboche (1974) used the concept of remaining life and continuous damage. Among the more advanced models the one by Chaboche (1974) should be mentioned since it is the basis for more sophisticated cumulative fatigue damage models for metals (Lesne and Savalle, 1987; Chaboche and Lesne, 1988) as well as composite materials (Arnold and Kruch, 1991a,b). Chaboche (1974) proposed the following fatigue damage evolution equation

$$dD = D^{\alpha(\sigma_M,\bar{\sigma})}\left[\frac{\sigma_M - \bar{\sigma}}{M(\bar{\sigma})}\right]^{\beta} dN \tag{2.4}$$

where σ_M and $\bar{\sigma}$ are the maximum and mean stresses, respectively. The quantities α, β and M are material dependent parameters. Lesne and Savalle (1987) and Chaboche and Lesne (1988) expanded the earlier fatigue damage model by Chaboche (1974) based on experimental investigations and observations, and proposed an improved fatigue damage evolution equation as follows

$$\frac{dD}{dN} = \left[1-(1-D_F)^{\beta+1}\right]^{\alpha(\sigma_M,\bar{\sigma})} \times \left[\frac{\sigma_M - \bar{\sigma}}{M(\bar{\sigma})(1-D_F)}\right]^{\beta} \tag{2.5}$$

The definitions of the variables used were the same as before. They were able to obtain the number of cycles to failure by integration of equation (2.5) as follows

$$N_F = \frac{1}{(\beta+1)\left[1-\alpha(\sigma_M,\bar{\sigma})\right]}\left(\frac{\sigma_M - \bar{\sigma}}{M(\bar{\sigma})}\right)^{-\beta} \tag{2.6}$$

Numerical results using this improved model showed excellent agreement with experimental results for the fatigue damage evolution and accumulation.

Chow and Wei (1991) proposed a continuum damage mechanics model for fatigue failure of metallic materials under multi-axial loading conditions. This model is

different from previous continuum fatigue damage models (Lemaitre, 1984; Chaboche and Lesne, 1988) in that a damage effect tensor of order 2 is introduced to allow for anisotropic fatigue damage development. This second order damage effect tensor itself is a function of two internal state variables that are measured experimentally. Using the thermodynamic theory of irreversible processes with internal state variables they defined a fatigue damage dissipative potential as follows

$$\Phi_d(\boldsymbol{Y}, B) = Y_d^{\frac{1}{2}} - (B_0 + B(w)) \tag{2.7}$$

and a fatigue damage criterion as

$$F_d = Y_d^{\frac{3}{4}} - E^{\frac{1}{2}} \ (B_0 + B(w)) = 0 \tag{2.8}$$

from which a fatigue damage evolution equation is derived. In equations (2.7) and (2.8), B_0 and $B(w)$ are defined as the initial damage strain hardening threshold and the increment of the damage strain hardening threshold, respectively, and w represents the overall material damage. The expression for Y_d is given as

$$Y_d = \frac{1}{2} \boldsymbol{Y}^T : \mathbb{J} : \boldsymbol{Y} \tag{2.9}$$

where

$$\mathbb{J} = \begin{bmatrix} 1 & \gamma \\ \gamma & 1 \end{bmatrix} \tag{2.10}$$

and γ is defined as the damage evolution coefficient. The quantity $\boldsymbol{Y}$ represents the thermo-dynamical force conjugate to the damage variable $\boldsymbol{D}$, namely the damage strain energy release rate, which is defined as follows

$$\boldsymbol{Y} = -\frac{\partial W^e(\boldsymbol{\sigma}, \boldsymbol{D})}{\partial \boldsymbol{D}} \tag{2.11}$$

while W^e is the energy expression for the elastic energy of a damaged material. Furthermore they introduced the hypothesis that fatigue damage is a cumulative result of elastic and plastic damages. Following the theory of irreversible thermodynamics with respect to different state variables they derived an expression for the damage evolution per fatigue cycle where they distinguished the two cases of elastic and plastic damage as stated in their hypothesis. The two expressions are given for the case of elastic damage ($\sigma_{max} \leq \sigma_d$) as

$$\frac{d\boldsymbol{D}}{dN} = -\int_0^{\sigma_{max}-\sigma_{min}} \frac{\boldsymbol{Y}^T : \mathbb{J} : d\boldsymbol{Y}}{\frac{8}{3} E^{\frac{1}{2}} Y_d^{\frac{3}{4}} K(w)} \mathbb{J} : \boldsymbol{Y} \tag{2.12}$$

and for the case of additional plastic damage $\sigma_{max} > \sigma_d$ as

$$\frac{d\boldsymbol{D}}{dN} = -\int_0^{\sigma_d-\sigma_{min}} \frac{\boldsymbol{Y}^T : \mathbb{J} : d\boldsymbol{Y}}{\frac{8}{3} E^{\frac{1}{2}} Y_d^{\frac{3}{4}} K(w)} \mathbb{J} : \boldsymbol{Y} - \int_{\sigma_d}^{\sigma_{max}} \frac{\boldsymbol{Y}^T : \mathbb{J} : d\boldsymbol{Y}}{\frac{8}{3} E^{\frac{1}{2}} Y_d^{\frac{3}{4}} \frac{\partial B}{\partial w}} \mathbb{J} : \boldsymbol{Y} \tag{2.13}$$

The two equations (2.12) and (2.13) reflect the proposed hypothesis that damage in a material occurs mainly due to two deformation states, the elastic and the plastic. The two states are distinguished by defining a damage stress threshold σ_d below which only elastic damage will develop and evolve, while additional plastic damage will develop and accumulate if the threshold stress is exceeded. Reasonable agreement was found when they compared their model results with experimental results of an aluminum alloy in the low cycle fatigue range. This was a first fairly successful attempt to tackle the fatigue problem and develop a fatigue damage model using potential theory and thermo-dynamical principles. The form of this model has been considered in the developed micro-mechanical fatigue damage model for MMC's since it includes the sophistication of anisotropic damage development and is based on thermo-dynamical principles. However for the case of composite materials only two damage variables that are proposed here do not suffice in order to describe the

damage development and evolution since a composite material, as modeled in this research, has to be considered as anisotropic.

When considering fatigue damage models for composite materials it is noticed that they are derived based on fatigue damage models developed for their isotropic counterparts. The use of this approach was mainly due to the fact that existing theoretical knowledge and sufficient experimental tests on composite materials were not readily available at the time. This route was chosen despite the well-known fact that the fatigue damage evolution and fatigue failure of composite materials is quite different from that of isotropic materials, such as the base matrix material. With improvement in theoretical knowledge on composite materials and experimental equipment a lot of studies have been conducted involving fatigue life and residual strength degradation, modulus degradation and residual life theories (Woodford, 1973; Johnson, 1983; Johnson and Wallis, 1984a; Charewicz and Daniel, 1985; Lemaitre, 1985a; Johnson, 1989; Russ et al., 1991; Johnson et al., 1993; Nicholas et al., 1993). However, it was soon recognized from the obtained models that the material structure of such composites has to be included in the development of fatigue damage models for composite materials in order to arrive at more feasible and reliable models (Hashin, 1985; Hwang and Han, 1986b).

A first cumulative fatigue damage model for composite materials was introduced by Owen and Howe (1972). Their model was developed as a stress independent model for a glass-reinforced plastic after studying debonding and resin cracking during fatigue loading. Their final conclusion was that resin cracking could be considered as damage in the composite and they proposed an expression for the damage variable

D as follows

$$D = B\left(\frac{n}{N}\right) - C\left(\frac{n}{N}\right)^2 \tag{2.14}$$

where B and C are material constants to be determined experimentally. It should be noted that the presented model predicts the same multi-stress level fatigue life with *Palmgren-Miner*'s rule since it is a stress-independent model.

Subramanyan (1976) proposed a non-linear cumulative fatigue damage model which was based on the S-N-curve, the fatigue limit and an isodamage line in the S-N-curve. Using these quantities he presented a final expression for the damage as follows

$$D = \left(\frac{\log N_k - \log N}{\log N_k - \log n}\right) \tag{2.15}$$

where N, n and N_k are defined as the number of cycles to failure at the current stress level, the number of current applied cycles, and the fatigue limit, respectively. This model was later improved by Srivatsavan and Subramanyan (1978) by including the effect of fatigue limit reduction.

Lemaitre and Plumtree (1979) proposed a cumulative creep-fatigue damage model for metals where a distinction is made for stress and strain controlled experiments. An expression for fatigue damage in case of a one-level fatigue loading was given as

$$D = 1 - \left(1 - \frac{n}{N}\right)^a \tag{2.16}$$

where

$$a = \frac{1}{p+1} \quad \text{(for strain-controlled test)}$$

$$a = \frac{1}{c+p+1} \quad \text{(for stress controlled test)}$$

p = material constant

c = material constant; maybe obtained from the S-N-curve

and n and N defined as the number of current applied cycles and the number of cycles at the fatigue limit, respectively. In the case of multi-stress level fatigue loading an expression for the damage in each loading block is defined as

$$D_{i+1} = 1 - \left[(1 - D_i)^{\frac{1}{a}} - \left(\frac{n}{N}\right)\right]^a \tag{2.17}$$

where n, N, D_i and D_{i+1} are the number of loading cycles in the current loading block, the number of cycles at the fatigue limit for the applied stress of the loading block, the accumulated fatigue damage at the beginning of the current loading block, and the accumulated fatigue damage at the end of n cycles in the current loading block, respectively. Upon close examination of the latter equation (2.17) it may be observed that an expression for the remaining life in case of multi-stress level fatigue is obtained as

$$n_{k+1} = N_{k+1}\left(1 - \sum_{i=1}^{k} \frac{n_k}{N_k}\right) \tag{2.18}$$

which is identical to the expression obtained from the *Palmgren-Miner* rule (equation 2.3) for multi-stress level fatigue loading.

Fong (1982) proposed a non-linear fatigue damage evolution equation as

$$\frac{dD}{dx} = k\,D + k' \tag{2.19}$$

where k, k' are material constants to be evaluated. He assumed that

- the damage function is a function of the normalized cycle ratio $\frac{n}{N}$ only, and that
- damage evolves linearly based on the current value of D.

Upon solving of equation (2.19) together with the initial conditions $D = 0, x = \frac{n}{N} = 0$ and the failure conditions $D = 1, x = \frac{n}{N} = 1$, the following expression for damage

maybe obtained as

$$D = \frac{e^{kx} - 1}{e^{k} - 1} \qquad k \neq 0 \tag{2.20}$$

$$D = x \qquad k = 0 \tag{2.21}$$

Throughout his research, Fong (1982) tried to match his model with physical quantities such as residual strength, number of debonded fibers, maximum crack length in the matrix, however with varying success.

Hashin (1985) compared cumulative fatigue damage models for composite materials subjected to variable amplitude cyclic loading. He showed that although both, the residual life and residual strength methods, use different approaches, they finally result in the same expression for a damage accumulation model which is equal to the well-known *Palmgren-Miner*'s rule. Hence he concluded that residual life and residual strength cumulative damage theories are completely equivalent. Further he pointed out that none of the presented models is capable of predicting appropriate experimental results accurately because the fatigue phenomenon is a random process. Hence it can not be described deterministically. To remedy this problem he suggested that a cumulative damage theory should predict the probability distribution function of the random lifetime variable rather than the lifetime itself.

Hwang and Han (1986b) introduced a new concept in the lifetime prediction of composite materials under constant amplitude cyclic loading which they titled "fatigue modulus concept". The fatigue modulus is defined as the ratio of the applied stress to the resultant strain at a specific cycle in the loading history and is expressed as

$$F(n,r) = \frac{\sigma_a}{\varepsilon_n} = \sigma_u \frac{r}{\varepsilon_n} \tag{2.22}$$

where

$F(n,r)$ = fatigue modulus at the n^{th} loading cycle

ε_n = resultant strain at the n^{th} loading cycle

σ_a = applied stress

r = ratio of applied stress σ_a to ultimate strength σ_u.

Using this concept together with the assumptions that:

1. the fatigue modulus degradation rate at an arbitrary cycle follows a power function of the fatigue cycle,

2. the fatigue life equation can be found as a function of the fatigue modulus, and

3. the fatigue life can be calculated using a strain failure criterion which states that final failure of the material occurs when the resultant strain reaches the static ultimate strain,

An expression for the fatigue lifetime of a composite material subjected to constant cyclic loading was found as

$$N = \left[\frac{F_0}{A}(1-r)\right]^{\frac{1}{c}} \tag{2.23}$$

where F_0 represents the fatigue modulus at $n = 0$, which is equal to the initial Young's modulus of the material, and c defined as a material constant to be determined experimentally. Comparison of their model with experimental data obtained from fatigue tests of glass epoxy composites showed reasonable agreement.

Based on the fatigue modulus concept (Hwang and Han, 1986b) and the resultant strain, Hwang and Han (1986a) proposed three different fatigue damage models to account for multi-stress level fatigue life in composite materials. In the first model they proposed that the variation of the fatigue modulus with respect to the number

of cycles. Hence it could be used as a measure for damage development. The fatigue damage could then be expressed as

$$D = \left(\frac{n}{N}\right)^C \tag{2.24}$$

where C is a constant to be determined experimentally (Hwang and Han, 1986a). The above equation has exactly the same form as the modified Palmgren-Miner's rule. In the second model fatigue damage at a specific cycle n was defined as the ratio of the resultant strain at the n^{th} cycle versus the failure strain

$$D = \frac{\varepsilon_n}{\varepsilon_f} \tag{2.25}$$

Using the equations developed earlier Hwang and Han (1986b) expressed damage as a function of the applied stress level r and the number of cycles n as

$$D = \frac{r}{\left(1 - \frac{n^C}{B}\right)}. \tag{2.26}$$

In the third model fatigue damage is defined as

$$D = \frac{\varepsilon_n - \varepsilon_0}{\varepsilon_f - \varepsilon_0} \tag{2.27}$$

with

$$\varepsilon_0 = \frac{\sigma_a}{F_0} \tag{2.28}$$

In this case damage is defined as a function of the resultant strain ε_n at the n^{th} cycle, the failure strain, ε_f, and the initial fatigue modulus, F_0. A final expression for the damage equation is then obtained as

$$D = \left(\frac{n}{N}\right)^C \left(\frac{B - N^C}{B - n^C}\right) \tag{2.29}$$

Comparing the developed models with experimental data for glass fiber epoxy composites they found that the third model shows better agreement for two-stress level fatigue life than other models. Finally they concluded that in order to explain the multi-stress level fatigue phenomena and to predict its life, the cumulative damage model approach is a feasible tool. Also to understand these phenomena they proposed to establish a cumulative damage model using physical variables rather than the number of cycles and the S-N curve.

Whitworth (1990) presented a nonlinear cumulative damage model for multi-stress level fatigue life time prediction of composite materials. He used the concept of stiffness degradation as a measure for the damage evolution with increasing number of cycles. For constant amplitude loading he proposed a damage function of the form

$$D = \left[\frac{H\,(1-\bar{S})^{a}}{(1-\bar{S}^{a})}\right]\frac{n}{N} \tag{2.30}$$

where a and H are material parameters independent of the applied stress level, and $\bar{S}$ is the normalized applied stress range with respect to the static ultimate strength. An undamaged state is then described by $D = 0$ whereas the damage at failure is described by $D = 1$. It maybe shown that for the case of $(a = 1)$ this model degenerates to the linear Miner damage rule which indicates that a linear degradation of the stiffness with respect to the applied number of cycles exists. Using the concept of equivalent damage (Leve, 1960) the model is expanded for fatigue life time prediction of composites under variable amplitude loading. In the equivalent cycle concept a particular number of cycles in a variable amplitude loading is transformed to an equivalent number of cycles in a reference stress-level such that the original and the transformed number of cycles render the same damage in the material at their respective cyclic stress level. This concept maybe simply expressed

for the case of a two-stress level loading as

$$D = \left[\frac{H(1-\bar{S}_1)^a}{1-\bar{S}_1^a}\right]\frac{n_1}{N_1} = \left[\frac{H(1-\bar{S}_2)^a}{1-\bar{S}_2^a}\right]\frac{n_{12}}{N_2} \tag{2.31}$$

where n_{12} represents that number of cycles at stress level σ_2 that would produce the same amount of damage as n_1 number of cycles at a stress level of σ_1. Hence the remaining lifetime n_2 at the second stress level maybe be determined as

$$n_2 = N_2 - n_{12} \tag{2.32}$$

The failure condition for this case is then given as

$$\sum_{i=1}^{2}\frac{r_i}{r_2}\frac{n_i}{N_i} = 1 \tag{2.33}$$

where

$$r_1 = \frac{(1-\bar{S}_1)^a}{1-\bar{S}_1^a} \quad \text{and} \quad r_2 = \frac{(1-\bar{S}_2)^a}{1-\bar{S}_2^a} \tag{2.34}$$

Upon use of equation (2.30) the failure condition for the case of a general variable amplitude loading is given as

$$\sum_{i=1}^{n}\frac{r_i}{r_n}\frac{n_i}{N_i} = 1 \tag{2.35}$$

which may be regarded as a modified form of Miner's rule. Reasonable agreement with experiments was obtained using the proposed model.

Arnold and Kruch (1991a,b) presented a phenomenological, isothermal transversely isotropic differential continuum damage mechanics (CDM) model for fatigue of uni-directional composites. This model is based on the CDM fatigue models for isotropic materials developed at ONERA (Office Nationale d'Etudes et de Recherches Aerospatiales) (Chaboche, 1981; Lesne and Savalle, 1987; Lesne and Cailletaud, 1987; Chaboche, 1987; Chaboche and Lesne, 1988; Chaboche, 1988a,b, 1989). A representative volume element (RVE, Figure 2.1) of a unidirectional fiber reinforced composite

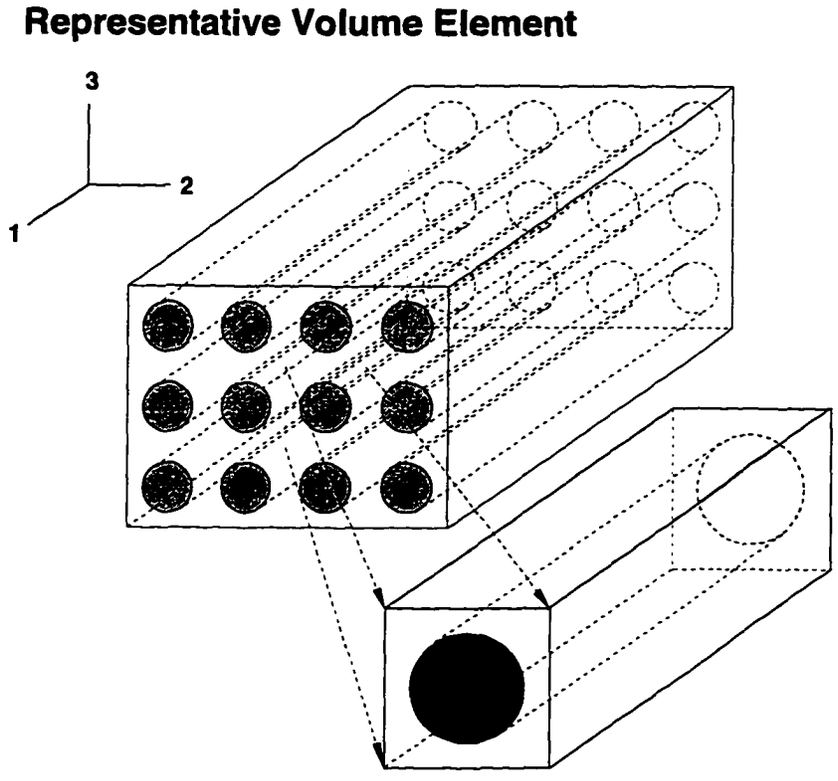

Figure 2.1: Definition of a representative volume element

material is used in the development of the model, assuming that this RVE includes a sufficient number of unit cells to allow the material to be considered as homogeneous (in a statistical sense) despite its heterogeneous nature. This allows one to directly relate the fatigue behavior and hence damage mechanisms of the composite material to that of the monolithic counterpart (the surface or interface of the fiber in the composite plays the role of the grain boundary in the monolithic material). However owing to the nature of composites it is necessary to include other damage modes, than matrix cracking and formation of micro cracks around the fiber, when describing damage, such as fiber breaking, fiber/matrix debonding and interlaminar cracking. Further it is postulated that damage in composites occurs in form of multiple cracking modes rather than as a single dominant crack, and that damage

develops along preferred directions, basically owing to the channeling effect of the fibers. Despite the fact that such direction-oriented damage modes would suggest a direction-dependent damage variable (Talreja, 1987) a scalar damage measure is assumed on the basis of the strong initial anisotropy of the material. The evolution of damage is considered as anisotropic and is associated with the preferred material direction. To model this damage, use is made of the definition of effective stress and remaining life as defined by Arnold and Kruch (1991b).

By introducing anisotropic damage surfaces and appropriate stress invariants that represent stress states that are likely to strongly influence the various damage modes in metallic composites they proposed the following multi-axial damage evolutionary law (similar to equation 2.5)

$$dD_F = \left[1 - (1 - D_F)^{\beta+1}\right]^{\alpha} \left[\frac{\hat{F}_m}{(1 - D_F)}\right]^{\beta} dN \tag{2.36}$$

with

$$\alpha = 1 - a\frac{< \hat{F}_{fl} - 1 >}{< 1 - \hat{F}_u >} \tag{2.37}$$

and

$$\sigma_{fl_L} = \sigma_{fl_L}(0)\,(1 - 3b'\,P) \tag{2.38}$$

$$M_L = M_{0_l}\,(1 - 3b\,P) \tag{2.39}$$

where a, b, b', $\sigma_{fl_L}(0)$, σ_{u_L} and M_{0_L} are material constants, and $< X >$ define the Macauley brackets. D_F represents the fatigue damage and the quantities $\hat{F}_{fl} - 1$, $1 - \hat{F}_u$ and $\hat{F}_m$ define the fatigue limit surface, the static fracture surface and the normalized stress amplitude, respectively. The parameter P represents the influence of the average hydrostatic stress

$$P = \sigma_{H,avg} = \frac{1}{2}\left\{max\left[\sigma_{ij}(t)\right] + min\left[\sigma_{ij}(t)\right]\right\} \tag{2.40}$$

For the case of a uni-axial stress state the following expression is used

$$\frac{d\,D_F}{d\,N} = \left[1-(1-D_F)^{\beta+1}\right]^{\alpha} \left(\mathfrak{S}_m\right)^{\frac{\beta}{2}} \left[\frac{S_{max}-\bar{S}}{(1-D_F)}\right]^{\beta} \tag{2.41}$$

and

$$\alpha = 1 - a\left\langle \frac{\left[\sqrt{\mathfrak{S}_{fl}}\,(S_{max}-\bar{S})-1\right]}{\left[1-\sqrt{\mathfrak{S}_u}\,S_{max}\right]} \right\rangle \tag{2.42}$$

where $\mathfrak{S}_m$, $\mathfrak{S}_{fl}$ and $\mathfrak{S}_u$ represent the expressions for the normalized stress amplitude, fatigue limit surface and static fracture surface, respectively. The quantities S_{max} and $\bar{S}$ are the normalized maximum (σ_{max}), and mean stresses (σ_{avg}), with respect to the ultimate static fracture stress in the fiber direction, σ_{u_L}.

An expression for the accumulation of damage during cycling is found to be

$$D_F = 1 - \left[1 - \left(\frac{N}{N_F}\right)^{\frac{1}{1-\alpha}}\right]^{\frac{1}{1-\beta}} \tag{2.43}$$

The fatigue lifetime of the material is then obtained as

$$N_F = \frac{1}{(1+\beta)\,(1-\alpha)} \left[\sqrt{\mathfrak{S}_m}(S_{max}-\bar{S})\right]^{-\beta} \tag{2.44}$$

The remaining life in the case of a two-stress level fatigue test may then be obtained as

$$N_2 = N_{F_2}\left[1 - \left(\frac{N_1}{N_{F_1}}\right)^{\frac{1-\alpha_2}{1-\alpha_1}}\right] \tag{2.45}$$

where N_{F_1} and N_{F_2} are the limit cycles for fatigue at stress levels σ_1 and σ_2, respectively, while N_1 and N_2 are the number of fatigue cycles for the appropriate stress levels and stress ratios. The results of a parametric study showed that the proposed model has predictive capabilities for anticipated qualitative trends in the

fatigue behavior of uni-axial metal matrix composites. Nevertheless it should be noticed that as result of its sophistication and complexity the proposed model has major drawbacks

- the expensive experimental setup to obtain the material parameters used in the model equations, and
- the employed scalar measure for the damage, which allows only for isotropic damage evolution.

Arnold and Wilt (1993) used the above fatigue damage model (Arnold and Kruch, 1991a,b) to develop a computational model for deformation and lifetime prediction of typical aerospace components. In the proposed decoupled analysis algorithm, the fatigue damage calculations were performed on a per cycle basis, whereas the stress computations of the performed 3-D finite element analysis were used. Fatigue damage related material properties were then calculated and fed back into the FE code to perform the analysis for the next cycle. This procedure was continued until final failure was reached. The results from this procedure were given qualitative as modified $S-N$ curves due to the lack of experimental data. In a subsequent publication Wilt and Arnold (1994) presented the coupled version of their algorithm. This time the fatigue damage analysis was included in the computational FE algorithm. The results of a sample analysis were presented in form of $S-N$ curves and damage distribution plots. Again no experimental results were available for comparison.

To complete this survey of fatigue damage models for composite materials it should be mentioned that in recent years the research on fatigue behavior and fatigue damage models for MMC's has focused on and included the high temperature effects, due to potential use of MMC's in high temperature environment such as gas

turbine engines (Neu and Sehitoglu, 1989a,b; Grady and Lerch, 1991; Russ et al., 1991; Jeng et al., 1992; Mirdamadi et al., 1993; Coker et al., 1993; Nicholas et al., 1993; Neu, 1993; Halford et al., 1993; Miller et al., 1993; Sanders and Mall, 1994; Neu and Roman, 1994; Hart and Mall, 1995; Nicholas, 1995). Various fatigue damage models considering mechanical fatigue at elevated temperatures as well as thermo-mechanical fatigue where the material is subjected additionally to temperature fluctuations on top of the mechanical fatigue loading, have been proposed (Neu and Sehitoglu, 1989b; Russ et al., 1991; Neu, 1993; Nicholas et al., 1993; Miller et al., 1993; Hart and Mall, 1995; Nicholas, 1995). Not only do the material properties change at elevated temperatures but damage mechanisms due to creep and oxidation might be detrimental to the material and have to be considered when modeling fatigue damage evolution (Neu and Sehitoglu, 1989a,b; Neu, 1993; Miller et al., 1993; Nicholas, 1995). As pointed out by Miller et al. (1993) only damage models which use a physically measurable quantity to describe and define damage, are versatile enough to model complex material behavior such as for the case of composites.

Based on the literature survey related to fatigue and fatigue damage modeling in composite materials it is found that with the advance in new technology and computational equipment the fatigue problem in composite materials maybe addressed by considering the micro structure of the composite material. In this sense it should be possible to model the fatigue behavior of composite materials by considering the material behavior at the constituent level. Upon use of appropriate homogenization procedures it should be possible to predict the response of the overall composite material subjected to fatigue loading based on the response of the individual constituents. This approach to model composite material behavior is known in the literature as micro-mechanical modeling and has been employed successfully for monotonic loading of composite materials.

Chapter 3

Micro-Mechanical Fatigue Damage Model

3.1 Introduction

Predicting the response of a material subjected to any kind of loading has been the focus of research ever since the subject of engineering was established. Sophisticated theories have been developed based on the phenomena observed in many experiments. With the advent of technology new methods have been developed for screening the material during the loading process and post-loading examinations, allowing to gain more in-depth knowledge on material behavior. A major effort in modeling constitutive equations for materials is dedicated today to the modeling of continuous material degradation by using the tool of continuous damage mechanics. Contrary to fracture mechanics, in continuous damage mechanics the material is assumed to retain its continuity hence no discontinuity in the deformation of adjacent material points is permitted. Actually continuous damage mechanics leads in the limit to fracture mechanics. Cracks, voids and cavities not considered and modeled explicitly as discontinuities in continuous damage mechanics theory, will grow eventually too big to be modeled under the umbrella of continuous damage mechanics, and hence fracture mechanics will take over as the modeling tool. Damage is defined as the mechanical degradation of material due to applied loading. Damage mechanics is the mathematical tool to describe the observed continuous physical phenomena of damage in the form of constitutive equations and damage evolution equations. Damage itself may be explained physically by means of micro-cracks, cavities and voids in the

material volume. As mentioned earlier in Chapter 1, the major problem in continuous damage mechanics consists of defining a proper mechanical variable which may be used in constitutive equations to describe the material behavior appropriately.

3.2 Continuous Damage Mechanics and the Damage Effect Tensor $\underline{M}$

Continuous damage mechanics was introduced in 1958 when Kachanov presented a theory to explain the creep failure of metals (Kachanov, 1958). In his work Kachanov used a continuity variable rather than a continuous damage variable to model the creep behavior of metals up to failure when subjected to sustained loading. Rabotnov (1968) revived and extended the idea of damage by Kachanov (1958) by introducing the effective stress concept through the definition of a scalar damage variable. In one dimension this concept is demonstrated best by considering Figure 3.1. As

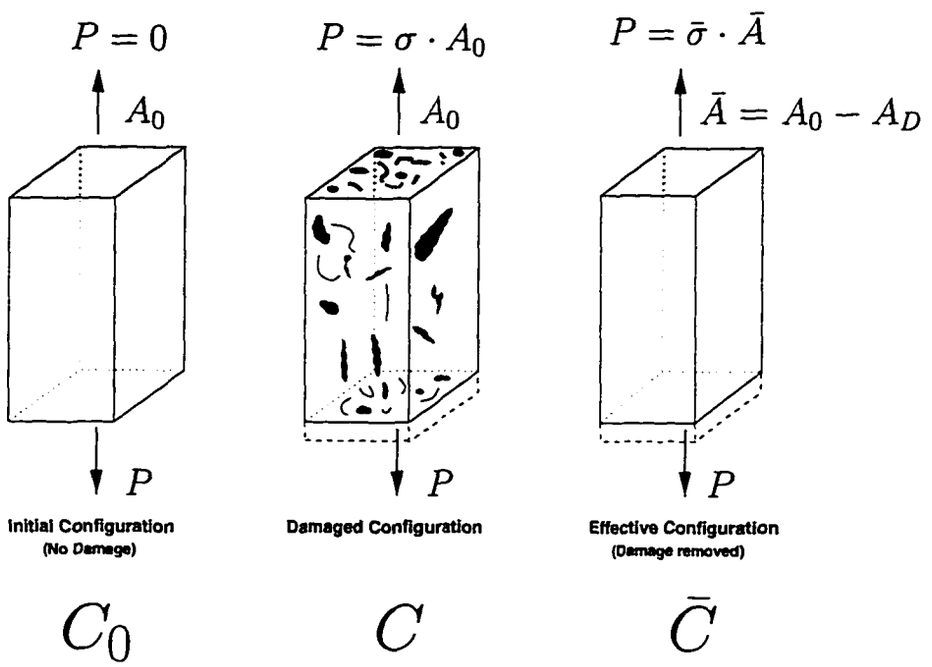

Figure 3.1: Schematical explanation of the Effective Stress Concept (Rabotnov, 1968)

observed in Figure 3.1 upon loading of a virgin (damage free) material (configuration C_0) and increasing the applied loading, the material encounters internal damage (configuration C) due to development of cracks, cavities and voids as a result of breakage of grains and loss of bondage at the grain boundaries of adjacent grains as well as due to expansion of existing initial cavities and voids as a result of developed defects. Upon hypothetical removal of the encountered damage, a damage free state (configuration $\bar{C}$), the so-called effective state, is obtained. In this effective state only the effective intact net cross-sectional area $\bar{A}$, which is the total cross-sectional area A_0 reduced by the damaged cross-sectional area A_D, resists the external applied load. Upon calculation of the engineering stress for the effective configuration $\bar{C}$ one obtains an expression for the effective stress based on the standard stress formula as

$$\bar{\sigma} = \frac{1}{1-D}\sigma \tag{3.1}$$

where σ is simply defined as the the ratio of the applied force P to the total cross-sectional area A_0. Here D represents the mechanical damage variable (Kachanov, 1958), in this case a scalar valued variable, which is simply defined as the ratio of the damaged cross-sectional area A_d to the original cross-sectional area A_0 in the one-dimensional case. The effective stress concept (Rabotnov, 1968) describes the degradation of the material with increasing load hence represents any kind of physical damage in the material. It is obvious from equation (3.1) that an undamaged material is represented by $D = 0$ while complete material failure in this context is given for $D = 1$. Numerous other more sophisticated damage variables have been proposed mostly in tensor form (Chaboche, 1978, 1979; Cordebois and Sidoroff, 1979; Sidoroff, 1979, 1980; Murakami and Ohno, 1980; Murakami, 1981, 1983, 1988) to allow for the representation of anisotropic damage development more appropriate

for the general type of applied loading and even more important for the material (e.g. composite material) used. In the following a symmetric second order damage tensor $\boldsymbol{\phi}$ (Murakami, 1981) is used in the development of the appropriate material and model equations.

Before continuing with the development of the effective stress concept to three dimensions, a few definitions for the used nomenclature are appropriate, in order to avoid misinterpretation and confusion in the subsequent formulations. In the following boldface letters define general tensors with respect to a fixed Cartesian system x_1, x_2, x_3. Regular boldface letters represent second order tensors (e.g. $\boldsymbol{A} \equiv A_{ij}$ or $\boldsymbol{\alpha} \equiv \alpha_{ij}$) while underlined boldface letters are used to represent fourth order tensors (e.g. $\underline{\boldsymbol{B}} \equiv B_{ijkl}$ or $\underline{\boldsymbol{\beta}} \equiv \beta_{ijkl}$). Furthermore one defines the following symbols for tensor operations

$$
\begin{aligned}
\boldsymbol{a} : \boldsymbol{b} &\equiv a_{ij}\, b_{ij} \\
\underline{\boldsymbol{A}} : \boldsymbol{a} &\equiv A_{ijkl}\, a_{kl} \\
\underline{\boldsymbol{A}}^T : \boldsymbol{a} &\equiv A_{ijkl}\, a_{ij} \\
\underline{\boldsymbol{A}} : \underline{\boldsymbol{B}} &\equiv A_{ijkl}\, B_{klmn} \\
\underline{\boldsymbol{A}}^T : \underline{\boldsymbol{B}} &\equiv A_{ijkl}\, B_{ijmn} \\
\underline{\boldsymbol{A}}^{-T} &\equiv \left(A^{-1}_{ijkl}\right)^T \\
\frac{\partial g}{\partial \boldsymbol{\phi}} &\equiv \frac{\partial g}{\partial \phi_{ij}} \\
\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{\phi}} &\equiv \frac{\partial a_{ij}}{\partial \phi_{kl}} \\
\frac{\partial \underline{\boldsymbol{A}}}{\partial \boldsymbol{\phi}} &\equiv \frac{\partial A_{ijkl}}{\partial \phi_{mn}} \\
\frac{\partial \underline{\boldsymbol{A}}}{\partial \boldsymbol{\phi}\, \partial \boldsymbol{\phi}} &\equiv \frac{\partial A_{ijkl}}{\partial \phi_{mn}\, \partial \phi_{pq}}
\end{aligned}
$$

where the superscripts "-1" and "T" indicate the inverse and the transpose of the appropriate tensor, respectively.

A direct extension of the effective stress concept in one dimension as described in equation (3.1) to three dimensions based on a second order symmetric damage tensor $\boldsymbol{\phi}$ would be given as (Murakami, 1988)

$$\bar{\sigma}_{ij} = (\delta_{ik} - \phi_{ik})^{-1} \sigma_{kj} \tag{3.2}$$

However this expression for the effective stress tensor $\bar{\boldsymbol{\sigma}}$ is asymmetric (Murakami, 1988) and hence not applicable to formulate constitutive and evolution equations for damaged materials. An appropriate as well as widely used symmetrization procedure (Murakami, 1988) is given as

$$\bar{\sigma}_{ij} = \frac{1}{2} \left[\sigma_{ik} (\delta_{kj} - \phi_{kj})^{-1} + (\delta_{ik} - \phi_{ik})^{-1} \sigma_{kj} \right] \tag{3.3}$$

Equation (3.3) may be rewritten in the following form as

$$\bar{\boldsymbol{\sigma}} = \underline{\boldsymbol{M}} : \boldsymbol{\sigma} \tag{3.4}$$

where $\underline{\boldsymbol{M}}$ represents the so-called fourth order damage effect tensor, while $\boldsymbol{\sigma}$ and $\bar{\boldsymbol{\sigma}}$ are the applied stress and the effective stress tensors, respectively. The damage effect tensor $\underline{\boldsymbol{M}}$ represents the mapping of the applied Cauchy stress $\boldsymbol{\sigma}$ onto the effective Cauchy stress $\bar{\boldsymbol{\sigma}}$ and is based on the above symmetrization procedure. Expanding equation (3.3) using a symbolic manipulation package (Mathcad 4.0) one obtains the individual components of the fourth order damage effect tensor $\underline{\boldsymbol{M}}$ by collecting appropriate terms for each stress component as

a) Effective stress component $\bar{\sigma}_{11}$

$$\bar{\sigma}_{11} = M_{11ij} \sigma_{ij} \tag{3.5}$$

where

$$
\begin{aligned}
M_{1111} &= \frac{1}{\Delta}(\chi_{22}\,\chi_{33} - \phi_{23}\phi_{32}) \\
M_{1112} &= \frac{1}{2\Delta}(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \\
M_{1113} &= \frac{1}{2\Delta}(\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}) \\
M_{1121} &= \frac{1}{2\Delta}(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \\
M_{1131} &= \frac{1}{2\Delta}(\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})
\end{aligned}
\tag{3.6}
$$

b) Effective stress component $\bar{\sigma}_{12}$

$$\bar{\sigma}_{12} = M_{12ij}\,\sigma_{ij} \tag{3.7}$$

where

$$
\begin{aligned}
M_{1211} &= \frac{1}{2\Delta}(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \\
M_{1212} &= \frac{1}{2\Delta}(\chi_{11}\,\chi_{33} + \chi_{22}\,\chi_{33} - \phi_{13}\,\phi_{31} - \phi_{23}\phi_{32}) \\
M_{1213} &= \frac{1}{2\Delta}(\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31}) \\
M_{1222} &= \frac{1}{2\Delta}(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \\
M_{1232} &= \frac{1}{2\Delta}(\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})
\end{aligned}
\tag{3.8}
$$

c) Effective stress component $\bar{\sigma}_{13}$

$$\bar{\sigma}_{13} = M_{13ij}\,\sigma_{ij} \tag{3.9}$$

where

$$
\begin{aligned}
M_{1311} &= \frac{1}{2\Delta}(\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23}) \\
M_{1312} &= \frac{1}{2\Delta}(\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21}) \\
M_{1313} &= \frac{1}{2\Delta}(\chi_{11}\,\chi_{22} + \chi_{22}\,\chi_{33} - \phi_{12}\,\chi_{21} - \phi_{23}\phi_{32})
\end{aligned}
\tag{3.10}
$$

$$M_{1323} = \frac{1}{2\Delta}(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32})$$
$$M_{1333} = \frac{1}{2\Delta}(\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})$$

d) Effective stress component $\bar{\sigma}_{21}$

$$\bar{\sigma}_{21} = M_{21ij}\,\sigma_{ij} \tag{3.11}$$

where

$$M_{2111} = \frac{1}{2\Delta}(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31})$$
$$M_{2121} = \frac{1}{2\Delta}(\chi_{11}\,\chi_{33} + \chi_{22}\,\chi_{33} - \phi_{23}\,\chi_{32} - \phi_{13}\phi_{31})$$
$$M_{2122} = \frac{1}{2\Delta}(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \tag{3.12}$$
$$M_{2123} = \frac{1}{2\Delta}(\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32})$$
$$M_{2131} = \frac{1}{2\Delta}(\phi_{23}\,\chi_{11} + \phi_{21}\phi_{13})$$

e) Effective stress component $\bar{\sigma}_{22}$

$$\bar{\sigma}_{22} = M_{22ij}\,\sigma_{ij} \tag{3.13}$$

where

$$M_{2212} = \frac{1}{2\Delta}(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31})$$
$$M_{2222} = \frac{1}{\Delta}(\chi_{11}\,\chi_{33} - \phi_{13}\phi_{31})$$
$$M_{2221} = \frac{1}{2\Delta}(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \tag{3.14}$$
$$M_{2223} = \frac{1}{2\Delta}(\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$
$$M_{2232} = \frac{1}{2\Delta}(\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21})$$

f) Effective stress component $\bar{\sigma}_{23}$

$$\bar{\sigma}_{23} = M_{23ij}\,\sigma_{ij} \tag{3.15}$$

where

$$
\begin{aligned}
M_{2313} &= \frac{1}{2\Delta}\left(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}\right) \\
M_{2321} &= \frac{1}{2\Delta}\left(\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23}\right) \\
M_{2322} &= \frac{1}{2\Delta}\left(\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21}\right) \\
M_{2323} &= \frac{1}{2\Delta}\left(\chi_{11}\,\chi_{22} + \chi_{11}\,\chi_{33} - \phi_{12}\,\phi_{21} - \phi_{13}\phi_{31}\right) \\
M_{2333} &= \frac{1}{2\Delta}\left(\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21}\right)
\end{aligned}
\tag{3.16}
$$

g) Effective stress component $\bar{\sigma}_{31}$

$$
\bar{\sigma}_{31} = M_{31ij}\,\sigma_{ij} \tag{3.17}
$$

where

$$
\begin{aligned}
M_{3111} &= \frac{1}{2\Delta}\left(\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}\right) \\
M_{3121} &= \frac{1}{2\Delta}\left(\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31}\right) \\
M_{3131} &= \frac{1}{2\Delta}\left(\chi_{11}\,\chi_{22} + \chi_{22}\,\chi_{33} - \phi_{12}\,\chi_{21} - \phi_{23}\phi_{32}\right) \\
M_{3132} &= \frac{1}{2\Delta}\left(\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}\right) \\
M_{3133} &= \frac{1}{2\Delta}\left(\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}\right)
\end{aligned}
\tag{3.18}
$$

h) Effective stress component $\bar{\sigma}_{32}$

$$
\bar{\sigma}_{32} = M_{32ij}\,\sigma_{ij} \tag{3.19}
$$

where

$$
\begin{aligned}
M_{3212} &= \frac{1}{2\Delta}\left(\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}\right) \\
M_{3222} &= \frac{1}{2\Delta}\left(\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31}\right) \\
M_{3231} &= \frac{1}{2\Delta}\left(\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}\right)
\end{aligned}
\tag{3.20}
$$

$$M_{3232} = \frac{1}{2\Delta}(\chi_{11}\chi_{22} + \chi_{11}\chi_{33} - \phi_{12}\phi_{21} - \phi_{13}\phi_{31})$$
$$M_{3233} = \frac{1}{2\Delta}(\phi_{32}\chi_{11} + \phi_{12}\phi_{31})$$

i) Effective stress component $\bar{\sigma}_{33}$

$$\bar{\sigma}_{33} = M_{33ij}\,\sigma_{ij} \tag{3.21}$$

where

$$\begin{aligned} M_{3313} &= \frac{1}{2\Delta}(\phi_{31}\chi_{22} + \phi_{21}\phi_{32}) \\ M_{3323} &= \frac{1}{2\Delta}(\phi_{32}\chi_{11} + \phi_{12}\phi_{31}) \\ M_{3331} &= \frac{1}{2\Delta}(\phi_{13}\chi_{22} + \phi_{12}\phi_{23}) \\ M_{3332} &= \frac{1}{2\Delta}(\phi_{23}\chi_{11} + \phi_{13}\phi_{21}) \\ M_{3333} &= \frac{1}{\Delta}(\chi_{11}\chi_{22} - \phi_{12}\phi_{21}) \end{aligned} \tag{3.22}$$

where Δ is the determinant of the tensor $(\delta_{ij} - \phi_{ij})$ given as

$$\begin{aligned} \Delta = \chi_{11}\chi_{22}\chi_{33} - \chi_{11}\phi_{23}\phi_{32} - \chi_{22}\phi_{13}\phi_{31} \\ - \chi_{33}\phi_{12}\phi_{21} - \phi_{12}\phi_{23}\phi_{31} - \phi_{13}\phi_{32}\phi_{21} \end{aligned} \tag{3.23}$$

The variables χ_{11}, χ_{22} and χ_{33} are defined as

$$\begin{aligned} \chi_{11} &= \delta_{11} - \phi_{11} \\ \chi_{22} &= \delta_{22} - \phi_{22} \\ \chi_{33} &= \delta_{33} - \phi_{33} \end{aligned} \tag{3.24}$$

All other tensor values M_{ijkl} not explicitly defined above are equal to zero. The above definition of the damage effect tensor $\underline{\underline{M}}$ represents the most general case of anisotropic damage. In the special case of no damage the damage effect tensor

reduces to the fourth order identity tensor $\underline{\underline{I}}$. This is readily shown by retaining only the nonzero terms of $\underline{\underline{M}}$ which are the components which include products of the form $\chi_{(ii)}\,\chi_{(jj)}$ (no sum on i or j). The determinant Δ reduces to unity for $\phi_{ij} = 0$. Hence the following non-zero components of $\underline{\underline{M}}$ remain as

$$\begin{aligned} M_{1111} &= 1 \\ M_{1212} &= 1 \\ M_{1313} &= 1 \\ M_{2121} &= 1 \\ M_{2222} &= 1 \\ M_{2323} &= 1 \\ M_{3131} &= 1 \\ M_{3232} &= 1 \\ M_{3333} &= 1 \end{aligned} \tag{3.25}$$

which comprise the components of the fourth order identity tensor $\underline{\underline{I}}$ defined as

$$I_{ijkl} = \delta_{ik}\,\delta_{jl} \tag{3.26}$$

Upon closer inspection of the damage effect tensor $\underline{\underline{M}}$ one may recognize that it may be expressed as the quotient of two tensor functions $\underline{\underline{F}}(\phi)$ and $g(\phi)$ as

$$M_{ijkl} = \frac{1}{2}\frac{F_{ijkl}}{g} \tag{3.27}$$

This form of $\underline{\underline{M}}$ is especially advantageous for the numerical implementation of the developed model where first and second order derivatives of $\underline{\underline{M}}$ with respect to damage tensor ϕ are required (refer to section 3.4.4). The partial derivatives of $\underline{\underline{M}}$ with respect to ϕ may then simply be calculated using the quotient rule of differential

calculus (Appendix C). For the first partial derivative one obtains

$$\frac{\partial M_{ijkl}}{\partial \phi_{mn}} = \frac{1}{2\,g^2}\left(g\,\frac{\partial F_{ijkl}}{\partial \phi_{mn}} - F_{ijkl}\frac{\partial g}{\partial \phi_{mn}}\right) \tag{3.28}$$

and while for the second partial derivative the following expression is obtained as

$$\begin{aligned}\frac{\partial^2 M_{ijkl}}{\partial \phi_{mn}\,\partial \phi_{pq}} = {} & \frac{1}{2\,g^3}\,\frac{\partial g}{\partial \phi_{pq}}\left(2\,F_{ijkl}\,\frac{\partial g}{\partial \phi_{mn}} - g\,\frac{\partial F_{ijkl}}{\partial \phi_{mn}}\right) \\ & + \frac{1}{2\,g^2}\left(g\,\frac{\partial^2 F_{ijkl}}{\partial \phi_{mn}\,\partial \phi_{pq}} - \frac{\partial F_{ijkl}}{\partial \phi_{mn}}\,\frac{\partial g}{\partial \phi_{pq}} - F_{ijkl}\frac{\partial^2 g}{\partial \phi_{mn}\partial \phi_{pq}}\right)\end{aligned} \tag{3.29}$$

Hence after determining the individual partial derivatives of the numerator tensor function F_{ijkl} and the denominator tensor function g, the final 6^{th} and 8^{th} order tensors $\frac{\partial M_{ijkl}}{\partial \phi_{mn}}$ and $\frac{\partial^2 M_{ijkl}}{\partial \phi_{mn}\partial \phi_{pq}}$, may readily be calculated, respectively. The components of the partial derivatives of $\underline{\underline{F}}$ and g with respect to $\boldsymbol{\phi}$ are given in detail in Appendix C.

3.3 Micro-Mechanical Modeling of Material Behavior

The conventional approach of describing the behavior of a material when subjected to loading consists of establishing constitutive equations for the material under consideration where this material is considered as a single entity. Hence material properties are established for the material on a macroscopic scale where no special consideration is given to the material substructure. The established material properties are considered as average properties for the material system under consideration and may be obtained using experimental methods or using the theory of mixtures or other appropriate homogenization procedures. For the latter two methods material properties for the material subgrades may be determined and then used in the chosen homogenization procedure to arrive at the global or overall material properties. These material properties are then used to describe the entire material behavior

when subjected to loading. Such an approach is called global or overall approach to modeling material behavior because no consideration is really given to the material behavior at the level of the individual constituents. In contrary to the global or overall approach the local or micro-mechanical approach tries to model the material behavior at the constituent level for each constituent individually. This is done by employing certain kinetic and kinematic boundary conditions at the common boundaries of the individual constituents in order to retain the deformation compatibility and force equilibrium internally in the material. Employing appropriate homogenization procedures it is then tried to model the global material behavior based on the material behavior of the individual constituents. In order to be able to model the material behavior of the individual constituents it is necessary to redistribute any external global applied loading or displacement to the individual constituents. Various methods which describe the distribution of global applied loading or displacement to the individual constituents have been developed, namely the self-consistent methods (Hill, 1965; Budiansky, 1965; Willis, 1977) and the Mori-Tanaka method (Mori and Tanaka, 1973; Benveniste, 1987; Norris, 1989; Lagoudas et al., 1991; Chen et al., 1992) which are based on the equivalent inclusion principle of Eshelby (1957). Among the afore-mentioned methods the Mori-Tanaka method is widely used mainly due its rather simple implementation as compared to the self-consistent method (Lagoudas et al., 1991), and hence is adopted in the following.

Employing these models it is possible to establish relations among the global (applied) and the local (constituent) stresses and strains based on the so-called stress and strain concentration tensors. Furthermore the concept of effective stress (equation 3.4) maybe employed at the local constituent level to each of the constituents

to yield the following relations as

$$\bar{\boldsymbol{\sigma}}^f = \underline{\boldsymbol{M}}^f : \boldsymbol{\sigma}^f \tag{3.30a}$$

$$\bar{\boldsymbol{\sigma}}^m = \underline{\boldsymbol{M}}^m : \boldsymbol{\sigma}^m \tag{3.30b}$$

where $\underline{\boldsymbol{M}}^f$ and $\underline{\boldsymbol{M}}^m$ are the fourth-order local damage effect tensors for the matrix and fiber material, respectively. The stress tensors $\bar{\boldsymbol{\sigma}}^f$ and $\bar{\boldsymbol{\sigma}}^m$ represent the local fiber and matrix stress tensors in the effective configuration, while $\boldsymbol{\sigma}^f$ and $\boldsymbol{\sigma}^m$ represent the fiber and matrix stress tensors in the damaged configuration, respectively. The local damage effect tensors $\underline{\boldsymbol{M}}^f$ and $\underline{\boldsymbol{M}}^m$ are derived on the same basis as described in section 3.2. However in this case the local fiber and matrix damage tensors $\boldsymbol{\phi}^f$ and $\boldsymbol{\phi}^m$ are used for the fiber and matrix damage effect tensors $\underline{\boldsymbol{M}}^f$ and $\underline{\boldsymbol{M}}^m$, respectively. A physical description of these damage tensors is given in Chapter 4.

3.3.1 Stress and Strain Concentration Tensors

The material under consideration is assumed to be made of a large number of aligned continuous cylindrical fibers embedded in a continuous matrix material typical of the micro-structure of the composite on average. The spatial arrangement of these embedded fibers is such that it may be regarded as homogeneous and transversely isotropic with respect to the fiber direction. Hence we may identify an element or substructure in the overall material structure which is repeated over and over again and making up the entire material structure. Such an element is schematically shown in Figure 2.1 and is called a representative volume element (RVE). For modeling purposes one considers only a single such element which consists of a single fiber surrounded by the matrix material, and assumes that it represents the entire material behavior adequately upon application of uniform elastic macroscopic loading.

Furthermore one considers uniform stresses and strains in the fibers and the matrix based on volume averages. The geometric distribution of the constituents in the representative volume element is described by means of fiber and matrix volume fractions, c_f and c_m, which simply represent the ratio of the initial fiber and matrix volume to the volume of entire representative volume element. It should be emphasized that initial in this case means that the material is in its virgin state hence it is assumed to be free of any damage. In the presence of damage the definition of the volume fraction as shown below is not realistic any more but they should be redefined and be replaced by the so-called effective volume fractions (Voyiadjis and Park, 1995b) as defined in Section 3.3.2. Nevertheless the current definition is valid for the virgin state and maybe used to define and calculate initial geometric and mechanical variables. The initial volume fractions are defined as

$$c_f = \frac{V_f}{V} \tag{3.31a}$$

$$c_m = \frac{V_m}{V} \tag{3.31b}$$

satisfying

$$c_f + c_m = 1 \tag{3.32}$$

Here V_f and V_m represent the volume of the fibers and the matrix in a representative volume element. Following the micro-mechanical model of Dvorak and Bahei-El-Din (1982), but solely for the elastic loading case since only high cycle fatigue is considered, the following relations between the elastic local average stresses and the elastic overall stress as well as between the elastic local average strains and the elastic global strain are valid

$$\boldsymbol{\sigma} = c_f \boldsymbol{\sigma}^f + c_m \boldsymbol{\sigma}^m \tag{3.33}$$

$$\boldsymbol{\varepsilon} = c_f \boldsymbol{\varepsilon}^f + c_m \boldsymbol{\varepsilon}^m \tag{3.34}$$

3.3.1.1 Stress Concentration Tensors

Using the Mori-Tanaka method referred to above, a direct relation between the global applied Cauchy stress and the local (constituent) Cauchy stresses is established as

$$\boldsymbol{\sigma}^f = \underline{\boldsymbol{B}}^f : \boldsymbol{\sigma} \tag{3.35a}$$

$$\boldsymbol{\sigma}^m = \underline{\boldsymbol{B}}^m : \boldsymbol{\sigma} \tag{3.35b}$$

where $\underline{\boldsymbol{B}}^f$ and $\underline{\boldsymbol{B}}^m$ represent the stress concentration tensors mapping the global applied Cauchy stress to the fiber and matrix, respectively. An expression for the fiber stress concentration tensor $\underline{\boldsymbol{B}}^f$ is given as (Lagoudas et al., 1991)

$$\underline{\boldsymbol{B}}^f = \left[\underline{\boldsymbol{I}}^{(4)} + c_m \underline{\boldsymbol{E}}^m : \left(\underline{\boldsymbol{I}}^{(4)} - \underline{\boldsymbol{S}}\right) : \left(\underline{\boldsymbol{C}}^f - \underline{\boldsymbol{C}}^m\right)\right]^{-1} \tag{3.36}$$

The matrix stress concentration tensor $\underline{\boldsymbol{B}}^m$ is obtained once the fiber stress concentration tensor $\underline{\boldsymbol{B}}^f$ is known by using the relation

$$c_f \underline{\boldsymbol{B}}^f + c_m \underline{\boldsymbol{B}}^m = \underline{\boldsymbol{I}}^{(4)} \tag{3.37}$$

and solving for $\underline{\boldsymbol{B}}^m$. In equations (3.36) and (3.37) the quantities $\underline{\boldsymbol{I}}^{(4)}$, $\underline{\boldsymbol{E}}^m$, $\underline{\boldsymbol{C}}^f$, $\underline{\boldsymbol{C}}^m$, $\underline{\boldsymbol{S}}$, c_m and c_f represent the fourth order identity tensor, defined as

$$I_{ijkl} = \delta_{ik}\,\delta_{jl} \tag{3.38}$$

the elasticity tensor of the matrix, the compliance tensor of the fiber, the compliance tensor of the matrix, the Eshelby tensor, the fiber and the matrix volume fractions, respectively. The Eshelby tensor $\underline{\boldsymbol{S}}$ based on the equivalent inclusion principle (Eshelby, 1957) is approximated for a cylindrical fiber embedded in an isotropic matrix according to Mura (1987). Expressions for the components of the Eshelby tensor are found in Appendix A.

Similarly a relationship between the global and local effective stresses is defined as

$$\bar{\sigma}^f = \underline{\bar{B}}^f : \bar{\sigma} \tag{3.39a}$$

$$\bar{\sigma}^m = \underline{\bar{B}}^m : \bar{\sigma} \tag{3.39b}$$

where $\underline{\bar{B}}^f$ and $\underline{\bar{B}}^m$ represent the effective stress concentration tensors for the fiber and matrix, respectively. Upon substitution of equation (3.4) for $\bar{\sigma}$ and equations (3.30) for $\bar{\sigma}^f$ and $\bar{\sigma}^m$ in equations (3.39) one obtains the following relation for the local damaged and effective stress concentration tensors as (Voyiadjis and Park, 1995b)

$$\underline{B}^f = \underline{M}^{-f} : \underline{\bar{B}}^f : \underline{M} \tag{3.40a}$$

$$\underline{B}^m = \underline{M}^{-m} : \underline{\bar{B}}^m : \underline{M} \tag{3.40b}$$

Solving for the local effective stress concentration tensors in equations (3.40) one obtains the following relation in terms of the local stress concentration tensors, $\underline{B}^f$ and $\underline{B}^m$, the local damage effect tensors, $\underline{M}^f$ and $\underline{M}^m$, and the global damage effect tensor $\underline{M}$ as (Voyiadjis and Park, 1995b)

$$\underline{\bar{B}}^f = \underline{M}^f : \underline{B}^f : \underline{M}^{-1} \tag{3.41a}$$

$$\underline{\bar{B}}^m = \underline{M}^m : \underline{B}^m : \underline{M}^{-1} \tag{3.41b}$$

3.3.1.2 Strain Concentration Tensors and Energy Equivalence Principle

When considering the mapping of the applied strain to the local constituent strains a similar procedure as outlined above for the mapping of the global applied stress to the local constituent stresses is followed. A relationship between the overall elastic

strain and local constituent strain is defined in the damaged configuration as

$$\varepsilon^f = \underline{A}^f : \varepsilon \tag{3.42a}$$

$$\varepsilon^m = \underline{A}^m : \varepsilon \tag{3.42b}$$

where $\underline{A}^f$ and $\underline{A}^m$ represent the elastic strain concentrations tensors for the fibers and the matrix, respectively. For the effective (fictitious undamaged) configuration one obtains

$$\bar{\varepsilon}^f = \bar{\underline{A}}^f : \bar{\varepsilon} \tag{3.43a}$$

$$\bar{\varepsilon}^m = \bar{\underline{A}}^m : \bar{\varepsilon} \tag{3.43b}$$

However in order to establish a direct relationship between the effective elastic strains and the damaged elastic strains in the constituents as well as the overall composite material, similar to the effective stress concept, one has to take recourse to the energy equivalence principle (Cordebois and Sidoroff, 1979; Sidoroff, 1980). The principle of energy equivalence is used instead of the proposed strain equivalence principle (Lemaitre, 1971) since the latter one will produce an asymmetric stiffness tensor for the general case of anisotropic damage in the material (Cordebois and Sidoroff, 1979). The principle of energy equivalence simply states that the expression for the complementary elastic energy of a damaged material has the same form as that of an undamaged material except that the stress tensor in the latter one is replaced by the effective stress tensor in the energy expression.

The complementary elastic energy of an undamaged material is defined as

$$V^e(\sigma, \phi = 0) = \frac{1}{2}\sigma : \underline{C} : \sigma \tag{3.44}$$

where $\underline{C} = \underline{E}^{-1}$ is defined as the undamaged elastic compliance tensor. For a damaged material the elastic complementary energy is then given as

$$V^e(\sigma,\phi) = V^e(\bar{\sigma},\mathbf{0}) = \frac{1}{2}\bar{\sigma} : \underline{C} : \bar{\sigma} = \frac{1}{2}\sigma : \left(\underline{M}^T : \underline{C} : \underline{M}\right) : \sigma \tag{3.45}$$

where σ, $\bar{\sigma}$ and ϕ are defined as the Cauchy stress, the effective Cauchy stress and the damage tensor, respectively. Using the energy expression V^e as the elastic potential an expression for the elastic strain is obtained as

$$\varepsilon = \frac{\partial V^e(\sigma,\mathbf{0})}{\partial \sigma} = \underline{C} : \sigma \tag{3.46}$$

for the undamaged configuration, and

$$\varepsilon = \frac{\partial V^e(\sigma,\phi)}{\partial \sigma} = \left(\underline{M}^T : \underline{C} : \underline{M}\right) : \sigma \tag{3.47}$$

in the damaged configuration. Expressing the elastic stress-strain relation in the damaged configuration similar to equation (3.46) as

$$\varepsilon = \bar{\underline{C}} : \sigma \tag{3.48}$$

an expression for the effective elastic compliance tensor $\bar{\underline{C}}$ is obtained as (Voyiadjis and Kattan, 1990)

$$\bar{\underline{C}} = \underline{M}^T : \underline{C} : \underline{M} \tag{3.49}$$

Using an expression for the elastic energy in the damaged material system defined as

$$\frac{1}{2}\varepsilon : \bar{\underline{C}}^{-1} : \varepsilon = \frac{1}{2}\bar{\varepsilon} : \underline{C}^{-1} : \bar{\varepsilon} \tag{3.50}$$

and substituting equation (3.49) yields the following expression as

$$\frac{1}{2}\varepsilon : \left(\underline{M}^{-1} : \underline{C}^{-1} : \underline{M}^{-1}\right) : \varepsilon = \frac{1}{2}\bar{\varepsilon} : \underline{C}^{-1} : \bar{\varepsilon} \tag{3.51}$$

Hence the elastic effective strain is then defined

$$\bar{\varepsilon} = \underline{M}^{-1} : \varepsilon \tag{3.52}$$

The same relations may now be established for the individual constituents as

$$\bar{\varepsilon}^f = \underline{\boldsymbol{M}}^{-f} : \varepsilon^f \tag{3.53a}$$

$$\bar{\varepsilon}^m = \underline{\boldsymbol{M}}^{-m} : \varepsilon^m \tag{3.53b}$$

Equations (3.53) together with equations (3.42), (3.43) and (3.52) yields

$$\underline{\bar{\boldsymbol{A}}}^f : \bar{\varepsilon} = \underline{\boldsymbol{M}}^{-f} : \underline{\boldsymbol{A}}^f : \underline{\boldsymbol{M}} : \bar{\varepsilon}^f \tag{3.54a}$$

$$\underline{\bar{\boldsymbol{A}}}^m : \bar{\varepsilon} = \underline{\boldsymbol{M}}^{-m} : \underline{\boldsymbol{A}}^m : \underline{\boldsymbol{M}} : \bar{\varepsilon}^m \tag{3.54b}$$

from which the expressions for the effective strain concentrations tensors are obtained as

$$\underline{\bar{\boldsymbol{A}}}^f = \underline{\boldsymbol{M}}^{-f} : \underline{\boldsymbol{A}}^f : \underline{\boldsymbol{M}} \tag{3.55a}$$

$$\underline{\bar{\boldsymbol{A}}}^m = \underline{\boldsymbol{M}}^{-m} : \underline{\boldsymbol{A}}^m : \underline{\boldsymbol{M}} \tag{3.55b}$$

3.3.2 Effective Volume Fractions

Recalling the definition of the volume fractions of the constituents (equations 3.31) it should be emphasized that the previous definition was based on an undamaged state. The derivation is based on the force equilibrium in a uni-directional composite material subjected to uni-axial loading in the fiber direction where none of the material constituents is damaged. Hence the entire cross-sectional area of the individual constituent is participating in resisting the applied loading. However, for the case that the constituents are damaged, the net cross-sectional area resisting the applied force is reduced due to the presence of damage. Therefore the load distribution in the material has changed and a load redistribution occurs in the material, such that the load previously carried by an undamaged portion of the cross-section, but which is damaged in the current configuration, is now carried by the remaining undamaged

portion of the cross-section. However, this mechanism will influence the original definition of the volume fraction hence the volume fraction will have to be redefined by considering the existing damage in the material. This leads to the definition of the so-called effective volume fractions (Voyiadjis and Park, 1995b) which incorporate the influence of the presence of damage in the material system. The expressions for the effective volume fractions are given as (for details on the derivation refer to Appendix B)

$$\bar{c}_m = \frac{1-\phi^m_{eq}}{(1-\phi^m_{eq}) + (1-\phi^f_{eq})\dfrac{c^0_f}{c^0_m}} \tag{3.56a}$$

$$\bar{c}_f = \frac{1-\phi^f_{eq}}{(1-\phi^f_{eq}) + (1-\phi^m_{eq})\dfrac{c^0_m}{c^0_f}} \tag{3.56b}$$

where ϕ^m_{eq} and ϕ^f_{eq} are the equivalent damage variables given as

$$\phi^m_{eq} = \frac{\|\boldsymbol{\phi}^m\|_2}{\|\boldsymbol{\phi}^m_{crit}\|_2} \tag{3.57a}$$

$$\phi^f_{eq} = \frac{\|\boldsymbol{\phi}^f\|_2}{\|\boldsymbol{\phi}^f_{crit}\|_2} \tag{3.57b}$$

with $\boldsymbol{\phi}^f_{crit}$ and $\boldsymbol{\phi}^m_{crit}$ defined as the critical damage tensor for the fiber and matrix, respectively. $\|\cdot\|_2$ denotes the operator symbol for the L_2-*norm* of the tensor variable enclosed. The critical damage tensor defines the limit values for the damage tensor components at which the material is failing due to rupture caused by inter-atomic decohesion (Lemaitre, 1985b; Lemaitre and Chaboche, 1990; Maugin, 1992). The critical values are smaller than the theoretical value of $\phi_{crit} = 1$, which is deduced from the one-dimensional representation of the effective stress concept (equation 3.1). According to Lemaitre and Chaboche (1990) the critical damage values actually

define the onset of macro-cracking in the material, hence fracture mechanics will replace continuum damage mechanics as the modeling tool. Lemaitre (1985b) and Lemaitre and Chaboche (1990) further stated that critical damage values vary for different materials as well as loading types, and they specified practical ranges for the critical damage values.

3.4 Fatigue Damage Model

Before deriving the anisotropic fatigue damage criterion and the appropriate damage evolution equation one needs to present the basic principles which underlie the derivation of the damage model. Furthermore it should be emphasized that in the following energy considerations only energy due to elastic deformation is considered since the following fatigue damage model is developed for high cycle fatigue where no plastic deformations at the scale of modeling occur.

3.4.1 Thermodynamic Principles

The first law of thermodynamics in solids states that the time rate of change of kinetic and internal energy of a body occupying the volume Ω is equal to the rate of work done on the body plus the rate of change per time unit in all other energies, such as heat. Mathematically this may be expressed as

$$\dot{K} + \dot{U} = M + Q \tag{3.58}$$

The individual quantities in the above equation are, the kinetic energy K defined as

$$K = \int_{\Omega} \frac{1}{2} \rho \boldsymbol{v} \cdot \boldsymbol{v} \, d\Omega \tag{3.59}$$

the internal energy U defined as

$$U = \int_{\Omega} \rho \epsilon \, d\Omega \tag{3.60}$$

the mechanical energy M defined as

$$M = \int_{\Omega} \rho \boldsymbol{b} \cdot \boldsymbol{v}\, d\Omega + \int_{\Gamma} \boldsymbol{\sigma}(\boldsymbol{n}) \cdot \boldsymbol{v}\, d\Gamma \tag{3.61}$$

and the heat energy Q defined as

$$Q = \int_{\Omega} \rho r\, d\Omega - \int_{\Gamma} \boldsymbol{q}(\boldsymbol{n})\, d\Gamma \tag{3.62}$$

In these equations the variables ρ, ϵ, $\boldsymbol{b}$, $\boldsymbol{v}$, $\boldsymbol{\sigma}(\boldsymbol{n})$, r and $\boldsymbol{q}(\boldsymbol{n})$ are defined as the mass density, the internal energy density per unit mass, the body force per unit mass, the velocity field, the surface traction on the surface element with outward normal $\boldsymbol{n}$, the internal body heat source per unit mass and the heat flux per unit time and unit surface area with outward normal $\boldsymbol{n}$. Γ represents the surface of the body occupying the volume Ω. Upon substitution of the these expressions and simplifications one obtains the equation for balance of linear momentum as

$$\rho \dot{v}_j = \sigma_{ij,i} + \rho b_j \tag{3.63}$$

and the equation of balance of energy as

$$\rho \dot{\epsilon} = \sigma_{ij}\, \dot{\varepsilon}_{ij} - \operatorname{div} \boldsymbol{q} + \rho r \tag{3.64}$$

where the velocity gradient $v_{j,i}$ has been replaced by the strain rate $\dot{\varepsilon}_{ij}$ since only infinitesimal strains are considered.

The second law of thermodynamics states that the total production of entropy is always larger or equal to zero. This may be formulated mathematically as (Eringen, 1967)

$$\dot{S} \geq \int_{\Omega} \frac{1}{T} \rho r\, d\Omega - \int_{\Gamma} \frac{1}{T} q_i\, n_i\, d\Gamma \tag{3.65}$$

where S represents the total entropy defined as

$$S = \int_{\Omega} \rho\, s\, d\Omega \tag{3.66}$$

with s as the entropy per unit mass and T the thermodynamic (absolute) temperature. Through simplification one arrives at the following inequality (local *Clausius-Duhem* inequality) as (Eringen, 1967)

$$\rho\, \dot{s} - \frac{\rho\, r}{T} + \text{div}\left(\frac{\boldsymbol{q}}{T}\right) \geq 0 \tag{3.67}$$

which specifies that the specific entropy production must always be non-negative at all times and points.

Introducing the *Helmholtz free energy* per unit mass $\mathcal{H}$, which is the sum of the internal energy and the irreversible heat energy, defined as

$$\mathcal{H} = \epsilon - s \cdot T \tag{3.68}$$

and taking the rate as

$$\dot{\mathcal{H}} = \dot{\epsilon} - s \cdot \dot{T} - \dot{s}\, T \tag{3.69}$$

together with equation (3.64) one may substitute into equation (3.67) to arrive at (Maugin, 1992)

$$\sigma_{ij}\, \dot{\varepsilon}_{ij} - \rho\left(\dot{\mathcal{H}} + s\, \dot{T}\right) - \frac{1}{T}(\boldsymbol{q} \cdot \boldsymbol{\nabla})\, T \geq 0 \tag{3.70}$$

If $\mathcal{H}$ is assumed to be a function of $\boldsymbol{\epsilon}$, T and the damage variable $\boldsymbol{\phi}$, one gets (Lubliner, 1990)

$$\dot{\mathcal{H}} = \frac{\partial \mathcal{H}}{\partial \varepsilon_{ij}}\, \dot{\varepsilon}_{ij} + \frac{\partial \mathcal{H}}{\partial T}\, \dot{T} + \frac{\partial \mathcal{H}}{\partial \phi_{ij}} \dot{\phi}_{ij} \tag{3.71}$$

and then the following inequality results as (Lubliner, 1990)

$$\left(\sigma_{ij} - \rho \frac{\partial \mathcal{H}}{\partial \varepsilon_{ij}}\right) \dot{\varepsilon}_{ij} - \rho \left(\frac{\partial \mathcal{H}}{\partial T} + s\right) \dot{T} - \rho \frac{\partial \mathcal{H}}{\partial \phi_{ij}} \dot{\phi}_{ij} - \frac{1}{T} (\boldsymbol{q} \cdot \boldsymbol{\nabla}) T \geq 0 \qquad (3.72)$$

Since this inequality must be valid for every time dependent deformation field and temperature distribution in Ω, one must assume for $\boldsymbol{\varepsilon}$ and T to be independent. Thus the following relationships are obtained

$$\sigma_{ij} = \rho \frac{\partial \mathcal{H}}{\partial \varepsilon_{ij}} \qquad \text{(stress relation)} \qquad (3.73)$$

$$s = -\frac{\partial \mathcal{H}}{\partial T} \qquad \text{(specific entropy)} \qquad (3.74)$$

and the dissipation inequality

$$-\rho \frac{\partial \mathcal{H}}{\partial \phi_{ij}} \dot{\phi}_{ji} - \frac{1}{T} (\boldsymbol{q} \cdot \boldsymbol{\nabla}) T \geq 0 \qquad (3.75)$$

In the case of isothermal conditions ($T = constant$) and no temperature gradients in the material ($\overrightarrow{\text{grad}}\, T = 0$) the following inequality involving only damage dissipation is obtained as

$$-\rho \frac{\partial \mathcal{H}}{\partial \phi_{ij}} \dot{\phi}_{ji} \geq 0 \qquad (3.76)$$

3.4.2 State Variables

According to the theory of thermodynamics of irreversible processes the state of a thermo-dynamical process may be uniquely described by a set of state variables which are characteristic for each physical phenomenon identified and to be analyzed. The choice of these state variables is vital for the success of modeling the physical phenomenon for which they have been defined. Processes described by means of these state variables are thermo-dynamically admissible if they satisfy the Clausius-Duhem inequality (equation 3.72) at any point in time of their evolution. The state variables

which are also denoted thermo-dynamic variables may be observable or internal. Observable variables in elasticity and damage mechanics have been identified as the total strain $\boldsymbol{\epsilon}$ and the temperature T. In the case of a pure elastic phenomenon the state depends uniquely on these variables at any instant of time. Phenomena such as damage and hardening are less obvious to identify visually and require the definition of internal (or hidden) state variables which represent the internal material state such as the configuration of micro-cracks and cavities or voids for the case of damage. In general there does not exist a direct way to measure such internal variables but since they describe the state of the material they are considered and treated as observable. No rule exists on the choice of such variables suited best to describe a certain phenomenon but experience, the ability do develop a physical feeling for the phenomenon or in some cases the type of application. The state variables chosen may be scalar, vector or tensor quantities for which temperature, stresses or strains are examples. Internal state variables do neither appear explicitly in equilibrium equations nor in the thermodynamic laws. Together with the definition of the state variables one must identify the thermo-dynamical conjugates or dual variables to the state variables which will then be derived by means of state potentials. The state potential is a scalar thermodynamic potential function of the state variables which has to satisfy the conditions of thermodynamic stability posted by the inequality derived from the second principle of thermodynamics (equation 3.72).

In the current situation of modeling fatigue damage development and evolution for high cycle fatigue loading the state variables identified and defined are the stress tensor $\boldsymbol{\sigma}$ and the damage tensor $\boldsymbol{\phi}$. The thermodynamic conjugates or dual variables are the strain tensor $\boldsymbol{\epsilon}$ and the thermodynamic force $\boldsymbol{Y}$, which is also called the elastic damage strain energy release rate (Cordebois, 1983). Evolution equations for internal

variables are derived from a dissipation potential defined as a scalar function of the dual variables.

3.4.3 The Thermodynamic Potential

To derive the thermodynamic conjugates or dual variables to the state variables a thermodynamic potential is defined such that it satisfies the *Clausius-Duhem* inequality (equation 3.72). For this purpose we choose a well-known potential function in form of the complementary elastic free energy or *Gibbs free energy* defined as (since we use a load controlled environment)

$$\rho \mathcal{G}(\boldsymbol{\sigma}, \boldsymbol{\phi}, T) = \boldsymbol{\sigma} : \boldsymbol{\varepsilon} - \rho \mathcal{H}(\boldsymbol{\varepsilon}, \boldsymbol{\phi}, T) \tag{3.77}$$

where $\mathcal{H}$ represents the *Helmholtz free energy* defined in equation (3.68). A suitable choice for the definition of the *Helmholtz free energy* is given through the total elastic energy stored in a damaged material system as

$$\rho \mathcal{H}(\boldsymbol{\varepsilon}, \boldsymbol{\phi}, T) = W^e(\boldsymbol{\varepsilon}, \boldsymbol{\phi}, T) \tag{3.78}$$

From this one obtains an expression for the complementary elastic energy in terms of the state variables $\boldsymbol{\sigma}$, $\boldsymbol{\phi}$ and T as

$$\rho \mathcal{G}(\boldsymbol{\sigma}, \boldsymbol{\phi}, T) = W^e(\boldsymbol{\sigma}, \boldsymbol{\phi}, T) \tag{3.79}$$

According to the hypothesis of generalized normality defined in thermodynamics the dual variables to the state variables are now defined as

$$\boldsymbol{\varepsilon} = \rho \frac{\partial \mathcal{G}}{\partial \boldsymbol{\sigma}} = \frac{\partial W^e(\boldsymbol{\sigma}, \boldsymbol{\phi}, T)}{\partial \boldsymbol{\sigma}} \tag{3.80}$$

and

$$\boldsymbol{Y} = \rho \frac{\partial \mathcal{G}}{\partial \boldsymbol{\phi}} = \frac{\partial W^e(\boldsymbol{\sigma}, \boldsymbol{\phi}, T)}{\partial \boldsymbol{\phi}} \tag{3.81}$$

For isothermal conditions the elastic energy W^e is independent of the temperature T hence the final expressions for the elastic strain $\boldsymbol{\varepsilon}$ and the thermodynamic force $\boldsymbol{Y}$ are given as

$$\boldsymbol{\varepsilon} = \frac{\partial W^e(\boldsymbol{\sigma}, \boldsymbol{\phi})}{\partial \boldsymbol{\sigma}} \tag{3.82}$$

$$\boldsymbol{Y} = \frac{\partial W^e(\boldsymbol{\sigma}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \tag{3.83}$$

Recalling the expression for the elastic energy in a damaged material system (equation 3.45) we obtain an expression for the elastic strain given by equation (3.47) as

$$\boldsymbol{\varepsilon} = \underline{\boldsymbol{M}}^T : \underline{\boldsymbol{C}} : \underline{\boldsymbol{M}} : \boldsymbol{\sigma} \tag{3.84}$$

and for the thermodynamic force or elastic damage energy release rate as (Voyiadjis and Park, 1995b)

$$\boldsymbol{Y} = \frac{\partial}{\partial \boldsymbol{\phi}} \left(\frac{1}{2} \boldsymbol{\sigma} : \underline{\boldsymbol{M}}^T : \underline{\boldsymbol{C}} : \underline{\boldsymbol{M}} : \boldsymbol{\sigma} \right) \tag{3.85}$$

which results ultimately in the following expression as

$$Y_{ij} = \frac{1}{2} \left(\sigma_{cd} \, C_{abpq} \, M_{pqkl} \, \sigma_{kl} + \sigma_{pq} \, M_{uvpq} \, C_{uvab} \, \sigma_{cd} \right) \frac{\partial M_{abcd}}{\partial \phi_{ij}} \tag{3.86}$$

3.4.4 Proposed Micro-Mechanical Fatigue Damage Model

3.4.4.1 Fatigue Damage Criterion

Based on thermo-dynamical principles and laws defined above a fatigue damage criterion is developed for high cycle fatigue loading. As a basis for the model development the anisotropic damage model for ductile materials by Stumvoll and Swoboda (1993) was followed. The model by Stumvoll and Swoboda (1993) has been adopted by Voyiadjis and Park (1995a) to model damage evolution in metal matrix composites by

introducing anisotropic damage parameters and evolution equations. Subsequently it was employed successfully to model anisotropic damage evolution for monotonic loading in metal matrix composites (Voyiadjis and Park, 1995a). For the development of the micro-mechanical fatigue damage model, the model by Voyiadjis and Park (1995a) is used where appropriate changes are made to include typical phenomena which are characteristic of fatigue damage development. Furthermore, the number of parameters for the damage criterion and hence damage evolution equation is reduced while other parameters are redefined to be able to model damage evolution due to high cycle fatigue loading. In the following it is understood that this model applies in principle to all constituents in the composite material individually to predict the fatigue damage development and evolution in the constituents. Furthermore it is obvious that appropriate material and model parameters have to be used for the individual constituents, mainly the fibers and matrix, and the interface where appropriate. Hence no special attention is given to either of the constituents unless it is vital for the understanding of the model or to avoid confusion.

Following the proposition of Chow and Lu (1989) one assumes that there exists a surface $g = 0$ in the stress space which separates the damaging state from the undamaging state. The surface $g = 0$ will be defined from now on as the damage surface. Such a damage surface may be defined as

$$g = \mathcal{F} - 1 \tag{3.87}$$

where $\mathcal{F}$ is defined as

$$\mathcal{F} = w_{ij}^{-1} w_{jk}^{-1} \left(Y_{kl} - \gamma_{kl}\right) \left(Y_{li} - \gamma_{li}\right) \tag{3.88}$$

The term $(Y_{kl} - \gamma_{kl})$ is introduced in order to represent the translation of the damage surface in the thermodynamic force space hence allowing to further tune the damage evolution for cyclic loading. This is similar to kinematic hardening in plasticity.

As mentioned above, the tensor $\boldsymbol{Y}$ represents the thermo-dynamical force conjugate to the damage variable $\boldsymbol{\phi}$ and is defined as given in equation (3.86). The second tensor quantity in parentheses, $\boldsymbol{\gamma}$, may in principle be compared to the backstress in plasticity theory, since it represents the center of the damage surface in the thermo-dynamical conjugate force space. The evolution of the movement of the center of the damage surface is defined as

$$\dot{\gamma}_{ij} = \mathrm{c}\,\dot{\phi}_{ij} \tag{3.89}$$

similarly to the evolution equation for the backstress in plasticity theory. The scalar quantity c is a material parameter and has to be determined to match experimental results.

The tensor quantity w_{ij} in equation (3.88) is defined to account for the anisotropic expansion of the damage surface as damage progresses (Voyiadjis and Park, 1995a) and is given as

$$w_{ij} = u_{ij} + V_{ij} \tag{3.90}$$

The tensor V_{ij} can be interpreted physically as the damage threshold tensor for the constituent material considered, hence defining the onset of damage. The tensor quantity $\boldsymbol{u}$ in the above equation is that part of the damage criterion which incorporates the uniform expansion of the damage surface during the evolution of damage and is defined similar to Voyiadjis and Park (1995a) as

$$u_{ij} = \delta_{ij}\,\lambda_{(i)} \left(\frac{\kappa}{\lambda_{(i)}}\right)^{\xi_{(i)}} \qquad \text{(no sum on } i\text{)} \tag{3.91}$$

The various quantities are defined as follows. The parameter κ is a scalar variable and defines the so-called "damage strengthening or hardening" parameter (Chow and Lu, 1989) which accounts for the increase in the initial damage threshold due

to micro-hardening in the material. Micro-hardening in this sense relates to the hardening process of the material at a very local level due to stress concentrations at material impurities (Suresh, 1991) not felt at the macro-scale, hence not noticeable in the form of macroscopic plastic deformation. Its evolution is based on the energy dissipated due to damage and is defined as

$$\kappa = \int_{\phi_1}^{\phi_2} \boldsymbol{Y} : d\boldsymbol{\phi} = \int_0^t \boldsymbol{Y} : \dot{\boldsymbol{\phi}} dt \tag{3.92}$$

The variable $\lambda_{(i)}$ relates to the elastic material properties of the constituent under consideration while the exponent $\xi_{(i)}$ in equation (3.91) is governing the evolution of damage based on the mean stress, the stress amplitude and the accumulated damage, hence will vary with respect to the number of cycles applied. The specific form of the variable ξ is addressed in chapter 4.1.

3.4.4.2 Damage Evolution

Following the definition of the fatigue damage surface, one has to determine the evolution equations of the internal variable of damage, $\boldsymbol{\phi}$. For this purpose one recalls the second principle of thermo-dynamics in the form of the *Clausius-Duhem inequality* as defined in equation (3.72), which states that dissipation must always be positive. Since one considers only high cycle fatigue loading where only elastic deformation takes place, no energy is dissipated due to plastic deformation but solely due to damage. Hence one may establish a dissipation potential incorporating energy dissipation solely due to damage. Such a potential may be expressed as

$$\Pi = \Pi^d = \boldsymbol{Y} : \dot{\boldsymbol{\phi}} \tag{3.93}$$

Since the dissipation process is stable, one needs to find the values for $\boldsymbol{\phi}$ which make equation (3.93) a minimum while the equation $g = 0$ is satisfied, since the surface g

(equation 3.87) separates a damaging stress state from a non-damaging stress state. Therefore one introduces the following potential

$$\Omega = \Pi^d - \dot{\Lambda}\, g \tag{3.94}$$

where $\dot{\Lambda}$ is a Lagrange multiplier. Applying the extremum principle with respect to the variable $\boldsymbol{Y}$ one obtains

$$\frac{\partial \Omega}{\partial Y_{ij}} = 0 \tag{3.95}$$

from which an expression for the damage increment is obtained as follows

$$\dot{\phi}_{ij} = \dot{\Lambda}\, \frac{\partial g}{\partial Y_{ij}} \tag{3.96}$$

Since rate dependency is not considered, equation (3.96) maybe expressed as follows

$$d\phi_{ij} = d\Lambda\, \frac{\partial g}{\partial Y_{ij}} \tag{3.97}$$

Equation (3.97) states that the damage increment $d\boldsymbol{\phi}$ is directed in the direction normal to the damage surface in the thermo-dynamical force space $\boldsymbol{Y}$ whereas the magnitude is determined by the scalar variable $d\Lambda$. Since the damage surface $g = 0$ separates damaging stress states from non-damaging stress states one may distinguish four different cases in the evolution state of the variable $\boldsymbol{\phi}$ when considering equation (3.97). It should be mentioned that a stress state beyond the surface $g = 0$, that is $g > 0$ is not stable since it does not represent a state of minimum energy. The four different states to be distinguished are as follows

$$g < 0 \tag{3.98a}$$

$$g = 0 \qquad \frac{\partial g}{\partial Y_{ij}}\, \dot{Y}_{ij} < 0 \tag{3.98b}$$

$$g = 0 \qquad \frac{\partial g}{\partial Y_{ij}}\, \dot{Y}_{ij} = 0 \tag{3.98c}$$

$$g = 0 \qquad \frac{\partial g}{\partial Y_{ij}} \dot{Y}_{ij} > 0 \tag{3.98d}$$

The only state of interest of these four is the last state described by equation (3.98d) since the states described by equations (3.98a - 3.98c) represent a non-damaging loading or unloading state, unloading from a damaging state, and a neutral loading state, respectively, all of which are characterized by $d\boldsymbol{\phi} = \mathbf{0}$. In contrary, the state described by equation (3.98d) describes a loading state from a damage state, hence damage will further evolve.

Recalling equation (3.97) one still has to determine the unknown scalar multiplier $d\Lambda$ controlling the magnitude of the damage increment $d\boldsymbol{\phi}$. In order to do this one has to recall that a damage state is always represented by satisfying $g = 0$. Hence upon further increase in damage this condition has to be satisfied and therefore the consistency condition is given as follows

$$dg = 0 \tag{3.99}$$

Since g itself is defined as a function of $\boldsymbol{\sigma}$, $\boldsymbol{\phi}$, κ and $\boldsymbol{\gamma}$, an expression for dg is obtained as

$$dg = \frac{\partial g}{\partial \boldsymbol{\sigma}} : d\boldsymbol{\sigma} + \frac{\partial g}{\partial \boldsymbol{\phi}} : d\boldsymbol{\phi} + \frac{\partial g}{\partial \kappa} d\kappa + \frac{\partial g}{\partial \boldsymbol{\gamma}} : d\boldsymbol{\gamma} = 0 \tag{3.100}$$

Recalling the definitions for κ (equation 3.92) and for $\dot{\boldsymbol{\gamma}}$ (equation 3.89), one may substitute appropriately into equation (3.100) such that

$$dg = \frac{\partial g}{\partial \boldsymbol{\sigma}} : d\boldsymbol{\sigma} + \frac{\partial g}{\partial \boldsymbol{\phi}} : d\boldsymbol{\phi} + \frac{\partial g}{\partial \kappa} \boldsymbol{Y} : d\boldsymbol{\phi} - \mathfrak{c} \frac{\partial g}{\partial \boldsymbol{Y}} : d\boldsymbol{\phi} = 0 \tag{3.101}$$

Replacing $d\boldsymbol{\phi}$ in equation (3.101) with equation (3.97) one obtains an expression for

$d\Lambda$ as

$$d\Lambda = -\frac{\dfrac{\partial g}{\partial \sigma_{kl}} d\sigma_{kl}}{\left(\dfrac{\partial g}{\partial \phi_{ij}} + Y_{ij} \dfrac{\partial g}{\partial \kappa} - \mathrm{c} \dfrac{\partial g}{\partial Y_{ij}} \right) \dfrac{\partial g}{\partial Y_{ij}}} \tag{3.102}$$

Backsubstitution of equation (3.102) into equation (3.103) yields an expression for the damage increment in terms of the stress increment as

$$d\phi_{mn} = -\frac{\dfrac{\partial g}{\partial \sigma_{kl}} \dfrac{\partial g}{\partial Y_{mn}} d\sigma_{kl}}{\left(\dfrac{\partial g}{\partial \phi_{ij}} + Y_{ij} \dfrac{\partial g}{\partial \kappa} - \mathrm{c} \dfrac{\partial g}{\partial Y_{ij}} \right) \dfrac{\partial g}{\partial Y_{ij}}} \tag{3.103}$$

or

$$d\phi_{ij} = \Psi_{ijkl}\, d\sigma_{kl} \tag{3.104}$$

where

$$\Psi_{ijkl} = -\frac{\dfrac{\partial g}{\partial Y_{ij}} \dfrac{\partial g}{\partial \sigma_{kl}}}{\left(\dfrac{\partial g}{\partial \phi_{rs}} + Y_{rs} \dfrac{\partial g}{\partial \kappa} - \mathrm{c} \dfrac{\partial g}{\partial Y_{rs}} \right) \dfrac{\partial g}{\partial Y_{rs}}} \tag{3.105}$$

A final expression for equation (3.105) is obtained by making use of the following partial derivatives

$$\begin{aligned} \frac{\partial g}{\partial Y_{rs}} &= \frac{\partial \mathcal{F}}{\partial Y_{rs}} \\ &= w_{ij}^{-1} w_{jr}^{-1} \left(Y_{si} - \gamma_{si} \right) + w_{kj}^{-1} w_{js}^{-1} \left(Y_{kr} - \gamma_{kr} \right) \end{aligned} \tag{3.106}$$

$$
\begin{aligned}
\frac{\partial g}{\partial \sigma_{ij}} &= \frac{\partial \mathcal{F}}{\partial Y_{rs}} \frac{\partial Y_{rs}}{\partial \sigma_{ij}} \\
&= \left[w_{ef}^{-1} w_{fp}^{-1} \left(Y_{qe} - \gamma_{qe} \right) + w_{kf}^{-1} w_{fq}^{-1} \left(Y_{kp} - \gamma_{kp} \right) \right] \\
&\quad \left\{ \frac{1}{2} \left[\bar{E}_{abmn}^{-1} M_{mnkl} \sigma_{kl} + \sigma_{mn} M_{uvmn} \bar{E}_{uvab}^{-1} \right] \frac{\partial M_{abij}}{\partial \phi_{rs}} \right. \\
&\quad \left. + \frac{1}{2} \left[\sigma_{cd} \bar{E}_{abmn}^{-1} M_{mnij} + M_{uvij} \bar{E}_{uvab}^{-1} \sigma_{cd} \right] \frac{\partial M_{abcd}}{\partial \phi_{rs}} \right\}
\end{aligned}
\tag{3.107}
$$

$$
\begin{aligned}
\frac{\partial g}{\partial \phi_{ij}} &= \frac{\partial \mathcal{F}}{\partial Y_{rs}} \frac{\partial Y_{rs}}{\partial \phi_{ij}} + \frac{\partial \mathcal{F}}{\partial \gamma_{rs}} \frac{\partial \gamma_{rs}}{\partial \phi_{ij}} \\
&= \left[w_{pq}^{-1} w_{qr}^{-1} \left(Y_{sp} - \gamma_{sp} \right) + w_{pq}^{-1} w_{qr}^{-1} \left(Y_{ps} - \gamma_{ps} \right) \right] \\
&\quad \times \left[\frac{1}{2} \left(\sigma_{cd} \bar{E}_{abef}^{-1} M_{efkl} \sigma_{kl} + \sigma_{ef} M_{uvef} \bar{E}_{uvab}^{-1} \sigma_{cd} \right) \frac{\partial^2 M_{abcd}}{\partial \phi_{rs} \partial \phi_{ij}} \right. \\
&\quad \left. + \frac{1}{2} \left(\sigma_{cd} \bar{E}_{abef}^{-1} \frac{\partial M_{efkl}}{\partial \phi_{ij}} \sigma_{kl} + \sigma_{ef} \frac{\partial M_{uvef}}{\partial \phi_{ij}} \bar{E}_{uvab}^{-1} \sigma_{cd} \right) \frac{\partial M_{abcd}}{\partial \phi_{rs}} \right] \\
&\quad - \mathbf{c} \left[w_{pq}^{-1} w_{qi}^{-1} \left(Y_{jp} - \gamma_{jp} \right) + w_{jp}^{-1} w_{pq}^{-1} \left(Y_{qi} - \gamma_{qi} \right) \right]
\end{aligned}
\tag{3.108}
$$

$$
\begin{aligned}
\frac{\partial g}{\partial \kappa} &= \frac{\partial \mathcal{F}}{\partial w_{rs}} \frac{\partial w_{rs}}{\partial \kappa} \\
&= -\left[w_{is}^{-1} w_{sj}^{-1} w_{jk}^{-1} \left(Y_{kl} - \gamma_{kl} \right) \left(Y_{li} - \gamma_{li} \right) \right. \\
&\quad \left. + w_{ij}^{-1} w_{jr}^{-1} w_{sk}^{-1} \left(Y_{kl} - \gamma_{kl} \right) \left(Y_{li} - \gamma_{li} \right) \right] \mathfrak{W}_{rs}
\end{aligned}
\tag{3.109}
$$

with

$$
Y_{rs} = \frac{1}{2} \left[\sigma_{cd} \bar{E}_{abpq}^{-1} M_{pqkl} \sigma_{kl} + \sigma_{pq} M_{uvpq} \bar{E}_{uvab}^{-1} \sigma_{cd} \right] \frac{\partial M_{abcd}}{\partial \phi_{rs}} \tag{3.110}
$$

$$
\mathfrak{W}_{rs} = \frac{\partial w_{rs}}{\partial \kappa} = \delta_{rs} \xi_{(r)} \left(\frac{\kappa}{\lambda_{(r)}} \right)^{\xi_{(r)}-1} \qquad \text{(no sum on } r\text{)} \tag{3.111}
$$

3.5 Constitutive Equations

The derivation of the elastic constitutive equation for the damaged composite material system is performed in two steps as shown in Figure 3.2. The first step involves

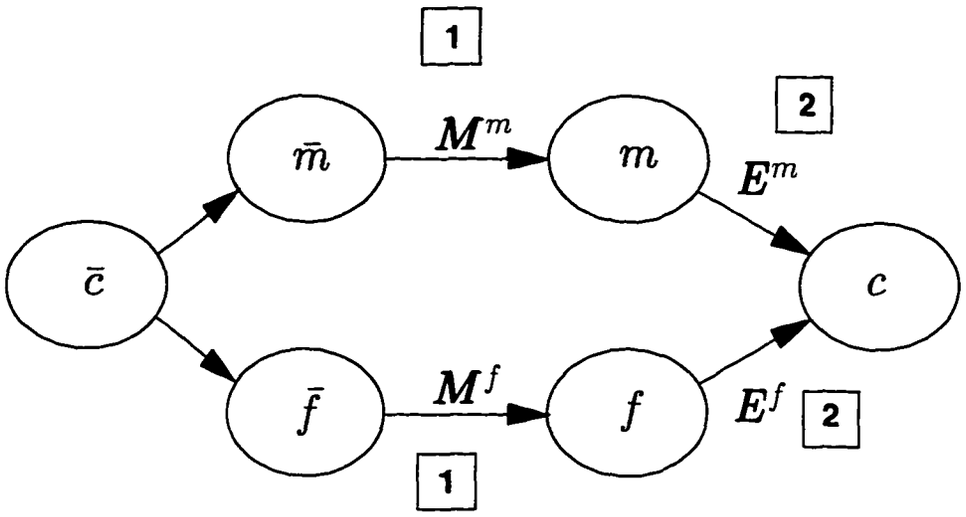

Figure 3.2: Schematic of micro-mechanical modeling

the derivation of the elastic constitutive equation for the individual constituents in the local damaged configurations for the fiber and the matrix based on the properties in the effective configuration. The second step is then used to combine the two constitutive equations for the fiber and the matrix in order to obtain the overall composite elastic constitutive equation. Only elastic deformations are considered here since only fatigue damage for high cycle fatigue loading is modeled. Since both constituents behave elastically under the applied load, one may develop the constitutive equation only for one of the constituents, and use a similar expression for the other constituent.

Considering for example the fiber, one obtains in the effective configuration $\bar{C}$ (Figure 3.1)

$$\bar{\sigma}^f = \underline{\bar{E}}^f : \bar{\varepsilon}^f \tag{3.112}$$

or

$$\bar{\varepsilon}^f = \underline{\bar{C}}^f : \bar{\sigma}^f \tag{3.113}$$

where $\underline{\bar{E}}^f$ and $\underline{\bar{C}}^f$ are the effective elasticity and compliance tensors for the fiber, where $\underline{\bar{C}}^f$ is the inverse of $\underline{\bar{E}}^f$ defined as

$$\underline{\bar{C}}^f = \underline{\bar{E}}^{-f} \tag{3.114}$$

Using equation (3.49) one is able to obtain the damaged fiber compliance tensor $\underline{C}^f$ in the damaged configuration as

$$\underline{C}^f = \underline{M}^{T^f} : \underline{\bar{C}}^f : \underline{M}^f \tag{3.115}$$

from which one may obtain the elasticity tensor of the fiber in the damaged configuration simply by taking the inverse of the compliance as defined in equation (3.115). Similarly one obtains the matrix compliance tensor in the damaged configuration as

$$\underline{C}^m = \underline{M}^{T^m} : \underline{\bar{C}}^m : \underline{M}^m \tag{3.116}$$

To obtain the overall composite behavior use is made of equations (3.33) and (3.34) for the damaged configuration. Then using the following relations

$$\varepsilon^f = \underline{C}^f : \sigma^f \tag{3.117}$$

$$\varepsilon^m = \underline{C}^m : \sigma^m \tag{3.118}$$

together with equations (3.35) one obtains the overall response for the composite elastic strain as

$$\begin{aligned} \varepsilon &= c_f \underline{C}^f : \underline{B}^f : \sigma + c_m \underline{C}^m : \underline{B}^m : \sigma \\ &= \left(c_f \underline{C}^f : \underline{B}^f + c_m \underline{C}^m : \underline{B}^m\right) : \sigma \end{aligned}$$

or

$$\varepsilon = \underline{C} : \sigma \tag{3.119}$$

where the elastic compliance of the overall composite material is now defined as

$$\underline{\boldsymbol{C}} = c_f \underline{\boldsymbol{C}}^f : \underline{\boldsymbol{B}}^f + c_m \underline{\boldsymbol{C}}^m : \underline{\boldsymbol{B}}^m \tag{3.120}$$

The tensor quantities $\underline{\boldsymbol{C}}^f$, $\underline{\boldsymbol{C}}^m$, $\underline{\boldsymbol{B}}^f$ and $\underline{\boldsymbol{B}}^m$ maybe obtained using equations (3.115), (3.116) and (3.40) based on the damage tensors $\boldsymbol{\phi}^f$ and $\boldsymbol{\phi}^m$ for the fiber and the matrix, respectively. Inspecting equations (3.40) more closely, one realizes that the overall damage effect tensor has not yet been obtained. To do so, one applies equation (3.33) to the effective configuration in order to obtain

$$\bar{\boldsymbol{\sigma}} = \bar{c}_f \bar{\boldsymbol{\sigma}}^f + \bar{c}_m \bar{\boldsymbol{\sigma}}^m \tag{3.121}$$

Here $\bar{c}_f$ and $\bar{c}_m$ represent the effective volume fractions as defined in equations (3.56). Using the following relations

$$\bar{\boldsymbol{\sigma}}^f = \underline{\boldsymbol{M}}^f : \boldsymbol{\sigma}^f = \underline{\boldsymbol{M}}^f : \underline{\boldsymbol{B}}^f : \boldsymbol{\sigma} \tag{3.122}$$

$$\bar{\boldsymbol{\sigma}}^m = \underline{\boldsymbol{M}}^m : \boldsymbol{\sigma}^m = \underline{\boldsymbol{M}}^m : \underline{\boldsymbol{B}}^m : \boldsymbol{\sigma} \tag{3.123}$$

to substitute in equation (3.121), one obtains a relation between the overall effective stress and overall applied stress by means of the overall damage effect tensor $\underline{\boldsymbol{M}}$ as

$$\bar{\boldsymbol{\sigma}} = \bar{c}_f \underline{\boldsymbol{M}}^f : \underline{\boldsymbol{B}}^f : \boldsymbol{\sigma} + \bar{c}_m \underline{\boldsymbol{M}}^m : \underline{\boldsymbol{B}}^m : \boldsymbol{\sigma} \tag{3.124}$$

or

$$\bar{\boldsymbol{\sigma}} = \underline{\boldsymbol{M}} : \boldsymbol{\sigma} \tag{3.125}$$

where $\underline{\boldsymbol{M}}$ is defined as

$$\underline{\boldsymbol{M}} = \bar{c}_f \underline{\boldsymbol{M}}^f : \underline{\boldsymbol{B}}^f + \bar{c}_m \underline{\boldsymbol{M}}^m : \underline{\boldsymbol{B}}^m \tag{3.126}$$

One is now able to determine the overall material response based on the response of the individual constituents.

3.6 Summary

A micro-mechanical damage theory based on thermo-dynamical principles is proposed for anisotropic materials subjected to high cycle fatigue loading. Only elastic deformations are considered. Modeling is performed using the effective stress concept (Rabotnov, 1968). Fatigue damage surfaces and appropriate damage evolution equations are established for the individual constituents in the effective configuration and then transferred to the real configuration by means of damage effect tensors. The stresses in the constituents are obtained from the external applied stresses using the Mori-Tanaka stress concentration tensors. Damage evolution in the overall composite is determined based on the damage evolution in the individual constituents. Physical phenomena such as the reduction in the net effective area/volume due to damage are considered by means of calculating effective volume fractions. Material and model parameters are defined based on the physical phenomena observed during fatigue tests.

In the following the proposed theoretical micro-mechanical fatigue damage model is implemented into a numerical simulation code in order to study the various parameters employed in the model. A suitable material system for this study is chosen based on the availability of experimental data to compare the numerical results with. Special attention is given to the determination of the material and model parameters. Valuable information with respect to fatigue damage damage development and evolution in materials is obtained from the literature in order to simulate properly the damage evolution process in the individual constituents. Parametric studies are performed in order to show the influence of the model and material parameters on the fatigue damage evolution process in the individual constituents. Special attention is given to the physical representation of the damage phenomena (effective volume

fractions, effective stresses) during the damage evolution process in the constituents as well as the overall composite.

Chapter 4
Model Implementation

The micro-mechanical fatigue damage model developed in the previous chapter is implemented into a numerical analysis code. In the development of the computer code all tensor equations are implemented as developed, with no simplifications such as the reduction of second order tensors to vectors, fourth order tensors to two-dimensional matrices, sixth order tensors to three-dimensional matrices or eighth order tensors to four-dimensional matrices. The full tensor analysis is chosen to allow for simpler and consistent coding, and reducing the risk of error during the transformation of the tensors, especially for higher order tensors. Furthermore, the availability of high performance computational equipment allowed to focus on the robust implementation of the model rather than to dedicate effort and time to the optimization and improvement of numerical efficiency of the code, which is left as a goal for future work. Prior to discussing some implementation aspects, such as the algorithm for the micro-mechanical modeling or the return to the damage surface in each increment during damage loading, procedures to obtain the material and model parameters are described below. A few numerical simulations of complete fatigue tests are then described in Chapter 5.

4.1 Material and Model Parameters

Material parameters used in the model (equations 3.89 to 3.91) are required for the individual constituent and must be determined appropriately for each constituent.

Following a common assumption for the constituents in composite materials, the constituents are considered isotropic. Hence in most cases the material properties are readily obtained from the pertinent literature or through the manufacturer of such composite materials, and need not be determined as part of the experimental procedures for the remaining material and model parameters. However, upon combining the constituents to form a composite material, one obtains a material which is considered as transversely isotropic with respect to the fiber direction. This is valid only for composite materials which contain continuous and aligned fibers in one direction, such as the composite material used in this work. Such a material, denoted as a uni-directionally fiber reinforced composite, is depicted in Figure 2.1. It is the basic building block for general composite materials which are made of stacks of such uni-directionally continuous fiber reinforced composite plates or laminae. To obtain a general composite material, the fiber directions in the individual laminae are rotated with respect to each other, allowing to tailor the material and its properties according to specific needs. Hence various forms of anisotropy maybe obtained. However, the material system in this work consists only of a uni-directionally continuous fiber reinforced metal matrix composite.

4.1.1 Material Properties

The material system used in this work consists of a uni-directionally continuous fiber reinforced metal matrix composite, identified as *SCS-6 / Ti-15-3*, which contains continuous Silicon Carbide fibers embedded in a Titanium alloy matrix and is a widely used metal matrix composite material system. The material properties for the individual constituents are obtained from the literature, namely from Johnson (1989), and are shown in Table 5.1. As stated before, the individual constituents are

considered as isotropic materials, rendering a transversely isotropic material upon combining them to form the metal matrix composite.

4.1.2 Damage Model Parameters

After obtaining the material parameters through the literature or the manufacturer, the next step involves the determination of the model parameters used in the model development of Section 3.4.4. Considering first the damage surface g as given in equation (3.87), one needs to establish the parameter V_{ij} in equation (3.90), defined as the damage threshold, which represents the onset of damage. The onset of damage is defined as that stage at which the state of stress causes the first damage to initiate. This is in general noticed by a change in the material behavior. Before and until damage initiates the material is in a virgin state and hence is considered isotropic. It should be emphasized that all these considerations are taken at the constituent level since the developed damage model represents a micro-mechanical damage model, which predicts the material behavior of the entire material system based on the material behavior of the individual constituents. Therefore damage onset in this context pertains to any of the constituents in the material system. Hence no distinction is made between the different constituents under consideration since the procedure applies to each of them. This requires that the damage initiation in each constituent has to be determined for each constituent individually. To determine the damage threshold for a given material one considers the following. Prior to damage initiation the material is free of damage, hence the damage tensor is given as

$$\phi_{ij} = 0 \tag{4.1}$$

Recalling the condition for damage to initiate or evolve as defined in equation (3.98), it is required for the state of stress to satisfy the damage surface such that

$$g = \mathcal{F} - 1 = 0 \tag{4.2}$$

where $\mathcal{F}$ is defined as given by equation (3.88). Upon closer inspection of equations (3.88), (3.90) and (3.91), one observes that the tensor quantity ω_{ij} is defined as a function of the damage hardening parameter κ which itself is dependent on the increment of damage variable, $d\phi_{ij}$, through equation (3.92). However, since no damage was incurred yet, the quantity κ takes a zero value, and so does the first term u_{ij} as given in equation (3.90) and defined in equation (3.91). Therefore the expression for ω_{ij} reduces in a damage free state to the following

$$\omega_{ij} = V_{ij} = \begin{bmatrix} V_{11} & 0 & 0 \\ 0 & V_{22} & 0 \\ 0 & 0 & V_{33} \end{bmatrix} \tag{4.3}$$

Based on the previous assumption of isotropy in the constituent materials, the above expression simplifies further to yield

$$\omega_{ij} = \begin{bmatrix} V & 0 & 0 \\ 0 & V & 0 \\ 0 & 0 & V \end{bmatrix} = \delta_{ij} V \tag{4.4}$$

where δ_{ij} represents the Kronecker delta. Considering equation (3.89) as the evolution equation for the parameter γ it is also clear that $\gamma_{ij} = 0$.

The last quantity defined as a function of the damage variable ϕ and which is needed in the calculation of the damage surface, is the thermo-dynamical force conjugate to the damage variable, $\boldsymbol{Y}$. Recalling the definition of $\boldsymbol{Y}$ as given in

equation (3.86), it is noticed that $\boldsymbol{Y}$ is a function of the damage variable $\boldsymbol{\phi}$ through the damage effect tensor $\underline{\boldsymbol{M}}$, which itself is a function of $\boldsymbol{\phi}$ as shown in Section 3.2. The damage effect tensor $\underline{\boldsymbol{M}}$ reduces to the fourth order identity tensor as

$$M_{ijkl} = \frac{1}{2}\left(\delta_{ik}\,\delta_{jl} + \delta_{il}\,\delta_{jk}\right) = I^{(4)}_{ijkl} \tag{4.5}$$

for the case that no damage exists, hence $\phi_{ij} = 0$. Assuming a uni-axial state of stress such as given by

$$\left[\sigma_{ij}\right] = \begin{bmatrix} \sigma_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

to determine the onset of damage, one may calculate Y_{ij} according to the definition given in equation (3.86). The partial derivative of M_{abcd} with respect to ϕ_{ij} maybe determined as outlined in equation (3.28). The necessary expressions are given in Appendix C. Upon closer inspection of equation (3.86) together with equations (4.6) and (4.5), one realizes that the expression for the thermo-dynamical force Y_{ij} in the case of a uni-axial state of stress simplifies to

$$Y_{ij} = \frac{1}{2}\left[\sigma_{11}\,\bar{C}_{ab11}\,\sigma_{11} + \sigma_{11}\,\bar{C}_{11ab}\,\sigma_{11}\right]\frac{\partial M_{ab11}}{\partial \phi_{ij}} \tag{4.7}$$

From the definition of the compliance tensor $\underline{\boldsymbol{C}}$ for an isotropic material

$$C_{ijkl} = \frac{(1+\nu)}{E}\left[-\frac{2\,\nu}{1+\nu}\,\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right] \tag{4.8}$$

it is clear, that only the terms $\bar{C}_{11ab}$ and $\bar{C}_{ab11}$ have non-zero entries, and are given as

$$\bar{C}_{1111} = \frac{1}{E} \tag{4.9a}$$

$$\bar{C}_{1122} = \bar{C}_{2211} = \bar{C}_{1133} = \bar{C}_{3311} = -\frac{\nu}{E} \tag{4.9b}$$

where E represents the Young's modulus for the material under consideration. Hence only the following partial derivatives $\frac{\partial M_{1111}}{\partial \phi_{ij}}$, $\frac{\partial M_{2211}}{\partial \phi_{ij}}$ and $\frac{\partial M_{3311}}{\partial \phi_{ij}}$ need to be considered in order to evaluate Y_{ij} for $\phi_{ij} = 0$. Out of these partial derivatives only the following one retains a non-zero value for $\phi_{ij} = 0$ as

$$\frac{\partial M_{1111}}{\partial \phi_{11}} = 1 \tag{4.10}$$

Hence the thermo-dynamical force $\boldsymbol{Y}$ for a uni-axial stress state and an undamaged material, where $\phi_{ij} = 0$, is defined as

$$\left[Y_{ij}\right] = \begin{bmatrix} Y_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{\sigma_{11}^2}{E} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

Backsubstitution of the inverse of ω_{ij} as defined in equation (4.4) together with equation (4.11) into equation (3.87) yields

$$g = \begin{bmatrix} \frac{1}{V^2} & 0 & 0 \\ 0 & \frac{1}{V^2} & 0 \\ 0 & 0 & \frac{1}{V^2} \end{bmatrix} \begin{bmatrix} Y_{11}^2 & 0 & 0 \\ 0 & 0 & \\ 0 & 0 & 0 \end{bmatrix} - 1 \tag{4.12}$$

Setting this expression to zero yields

$$g = \frac{Y_{11}^2}{V^2} - 1 = 0 \tag{4.13}$$

Substituting the expression for Y_{11} from equation (4.11) gives

$$g = \frac{\sigma_{11}^4}{E^2 V^2} - 1 = 0 \tag{4.14}$$

From this equation the material parameter V, namely the damage threshold, is determined as

$$V = \frac{\sigma_{11}^2}{E} \tag{4.15}$$

which represents twice the elastic energy in the system at the onset of damage. Upon experimental determination of the onset of damage by observing the change in material behavior as load increases, the parameter V maybe determined based on the applied stress and the Young's modulus of the material under consideration. A few sample analyses for different values of V for the fibers and the matrix are shown in Figures 4.1 and 4.2, in order to demonstrate the influence of V on the

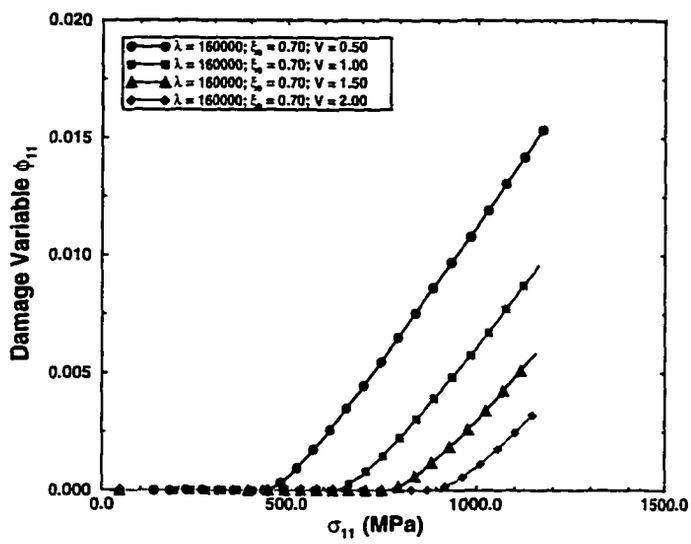

Figure 4.1: Damage initiation in the fiber with variation in V^f

damage initiation in the constituents. It is clear from Figures 4.1 and 4.2 that an increase in the damage threshold V of the material delays the initiation of damage. This is expected since a material with a higher value for V is stronger and therefore a higher energy is needed in order to break the internal material structure and/or expanding existing micro-cavities and micro-voids which is associated with damage development in a material.

Another parameter used in the model to be determined is the parameter λ as given in equation (3.91). It is defined as *Lamé's* constant in elasticity (Stumvoll and

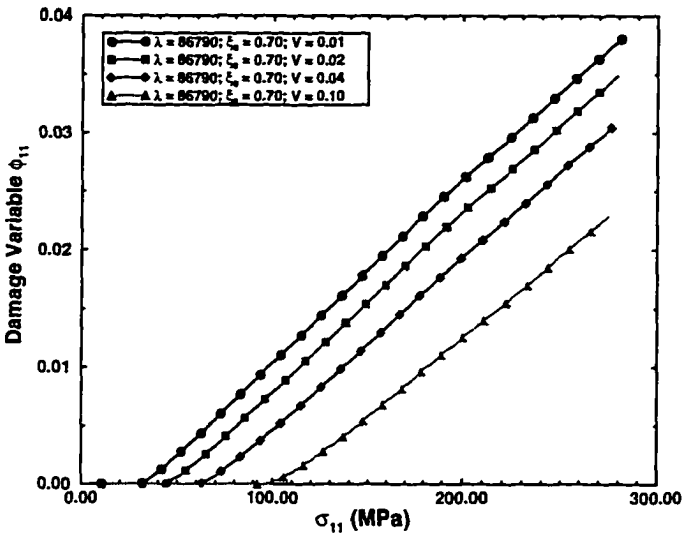

Figure 4.2: Damage initiation in the matrix with variation in V^m

Swoboda, 1993; Voyiadjis and Park, 1995b), which is defined for an isotropic material as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{4.16}$$

The last parameter to be determined is the exponent ξ as given in equation (3.91). This parameter is most influential in the damage evolution of the material. This observation is based on the results of a parametric study which are shown in Figures 4.3 and 4.4. For this study all parameters except ξ are kept constant. The parameter ξ is assigned different values, and then used to investigate the change in damage evolution. From Figures 4.3 and 4.4 it is clear that small changes in ξ will have a considerable influence on the damage evolution. However, as noticed from the results of a few numerical analyses using various values of ξ over a number of simulated fatigue cycles (Figure 4.5), it is observed that the damage evolution rate is decreasing with increasing number of cycles. This contradicts the experimentally observed

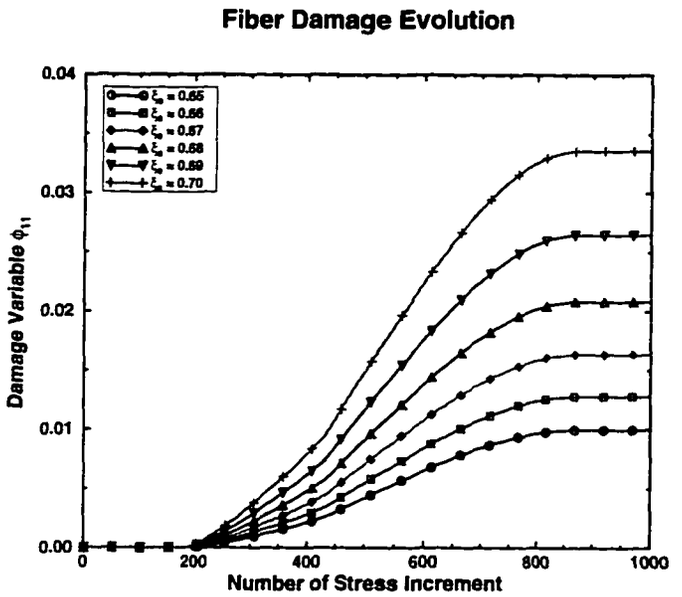

Figure 4.3: Variation in fiber damage evolution with respect to ξ

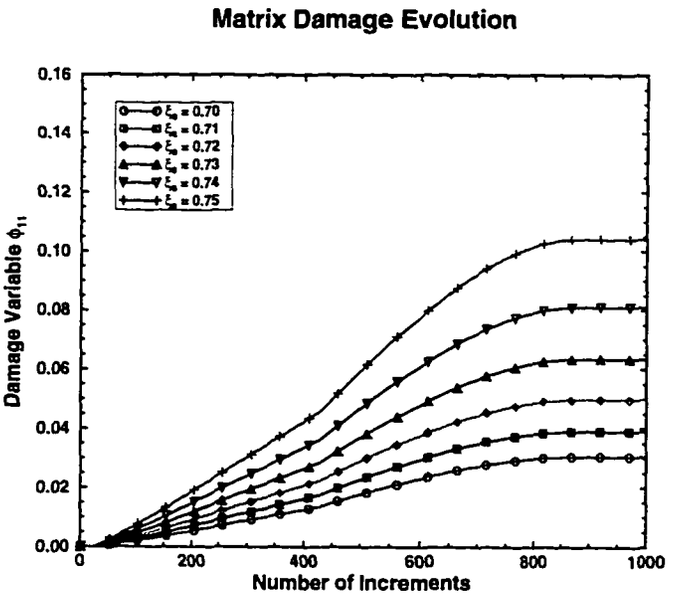

Figure 4.4: Variation in matrix damage evolution with respect to ξ

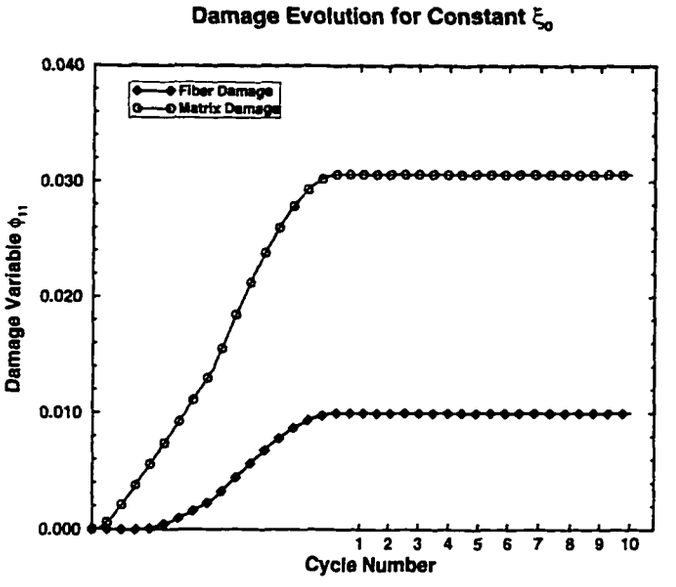

Figure 4.5: Variation of damage for constant ξ

behavior for damage evolution in materials subjected to high cycle fatigue as stated by Dieter (1988) and Suresh (1991). According to them a fairly constant damage evolution and propagation behavior is exhibited after the first few loading cycles for a large portion of the fatigue life.

This experimentally observed behavior led to the conclusion that the parameter ξ must not be a constant value but that it has to be a function of the number of applied cycles. To find a relation for ξ with respect to the number of applied cycles requires the experimental determination of a damage evolution curve, where the damage in the material is plotted versus the number of applied cycles. Due to the lack of such experimental data for metal matrix composites, such a curve is shown qualitatively in Figure 4.6. As indicated in this figure, there maybe in general 3 evolution phases to be distinguished (Dieter, 1988; Suresh, 1991). These phases are the initial damage phase (Phase I), the damage initiation and propagation phase (Phase II), and the

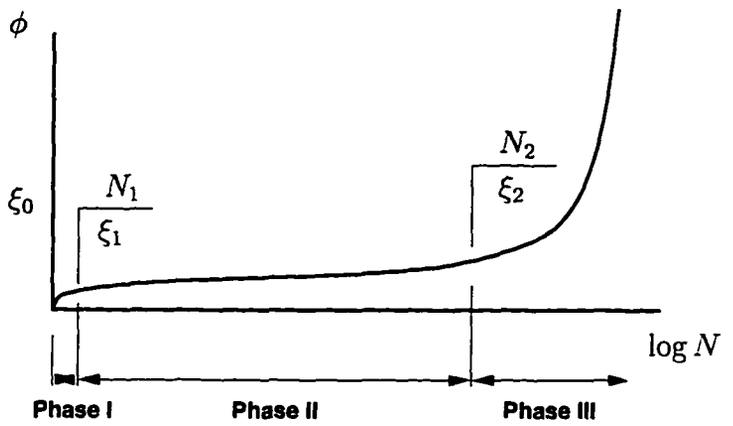

Figure 4.6: Qualitative $\phi - N$ curve

failure phase (Phase III). In Phase I, which takes a very short period of the entire fatigue life, mostly only a few cycles, the first cracks initiate due to micro-structural adjustment in the material structure. Phase II takes in general the major part of the fatigue life and is defined by a fairly constant crack initiation and propagation behavior (Dieter, 1988; Suresh, 1991). Especially for high cycle fatigue loading this region takes up a vast amount of the life time, before in Phase III the cracks start to propagate due to internal overload leading to failure of the material. Examples of such damage evolution curves for high cycle fatigue are shown in Figure 4.7 (Lemaitre and Chaboche, 1990) and Figure 4.8 (Lesne and Savalle, 1987) for the case of metals.

The last phase is in general identified by a highly non-linear damage evolution. Internally, at the transition from phase II to phase III, the transcrystalline cracking mode changes to an intercrystalline cracking mode (Lesne and Savalle, 1987).

The expression for ξ therefore should assure that the simulated damage evolution reflects this general behavior, where phase I is the least significant, especially for long fatigue life (high cycle fatigue). However it is retained for the purpose of generality. Failure in composite materials occurs in general due to overstress fracture of

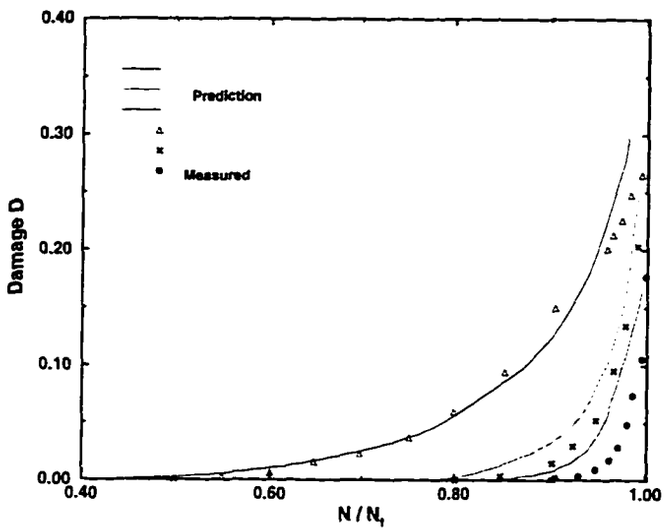

Figure 4.7: Experimentally obtained fatigue damage evolution for high cycle fatigue of metals (After Lemaitre and Chaboche, 1990)

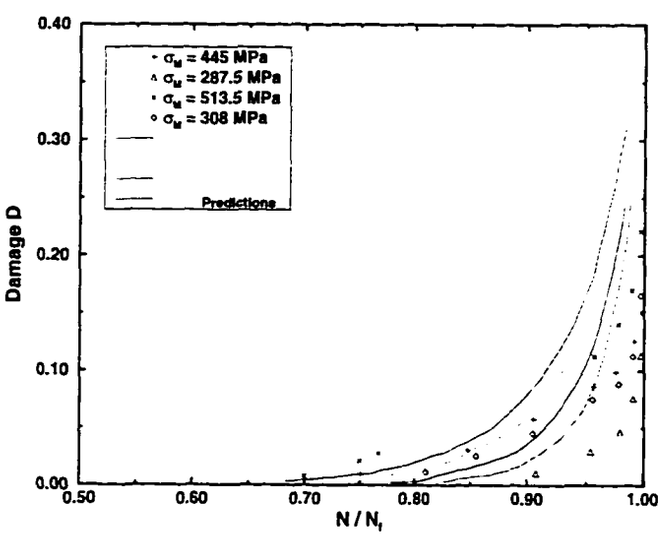

Figure 4.8: Experimentally determined fatigue damage evolution for high cycle fatigue of metals (After Lesne and Savalle, 1987)

the remaining undamaged constituent or due to debonding at the fiber-matrix interface. Overstress fracture occurs in composites with a strong interface which allows the cracks to develop in one constituent to propagate across the interface into the other one. This behavior is also called self-similar crack growth behavior since it resembles the crack growth mechanisms in pure metals (Johnson, 1989). In contrary to overstress fracture, failure due to debonding occurs in composites with very weak interfaces, where the cracks developed in a constituent propagate to the interface and then along the interface, leading ultimately to debonding and therefore separation of the fiber and the matrix. Such failure modes maybe simulated through the proper adjustment of the evolution equations of ξ. An experimental procedure needs to be used here in order to obtain the needed $\phi - N$ curves for metal matrix composites. This proposed procedure was successfully applied to for monotonic loading by Voyiadjis et al. (1993), Venson (1994) and Voyiadjis and Venson (1995).

Using a few such $\phi - N$ curves it is possible to obtain a relation for ξ with respect to the applied mean stress σ_{mean} while the stress ratio R remains constant (Figure 4.9), or with respect to the stress ratio R for a fixed mean stress σ_{mean} (Figure 4.10). For clarification, the stress ratio R is defined as the ratio of the minimum applied cyclic stress, σ_{min}, to the maximum applied cyclic stress, σ_{max}, in a fatigue cycle, and is given as

$$R = \frac{\sigma_{min}}{\sigma_{max}} \tag{4.17}$$

and shown in Figure 4.11. The mean stress σ_{mean} is defined as the arithmetic average of the maximum and the minimum applied stress. It is given as

$$\sigma_{mean} = \frac{1}{2} \left(\sigma_{max} + \sigma_{min} \right) \tag{4.18}$$

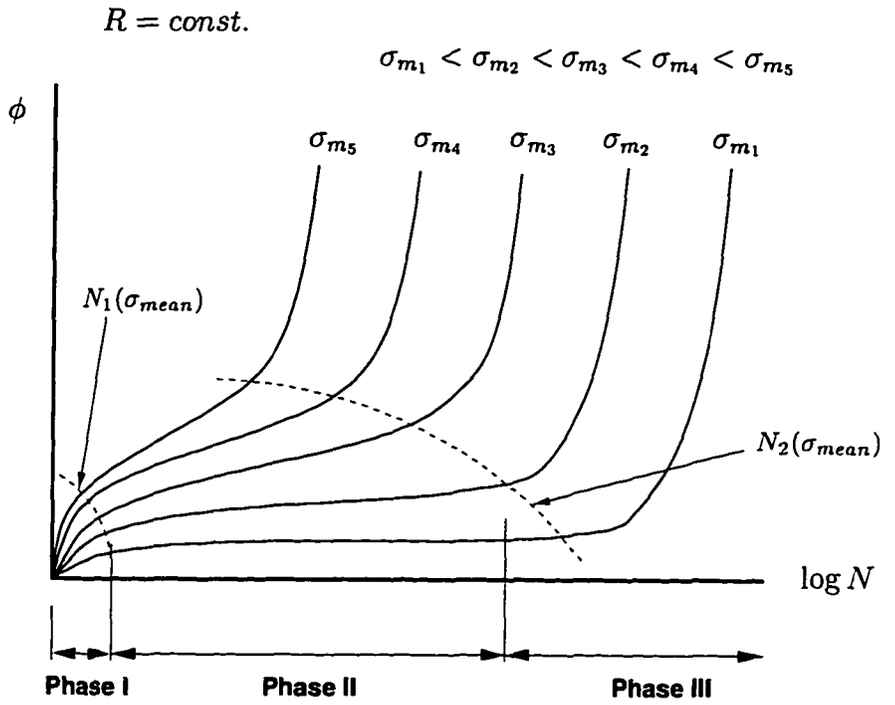

Figure 4.9: Fatigue damage evolution curves ($\phi - N$ curves) for constant R

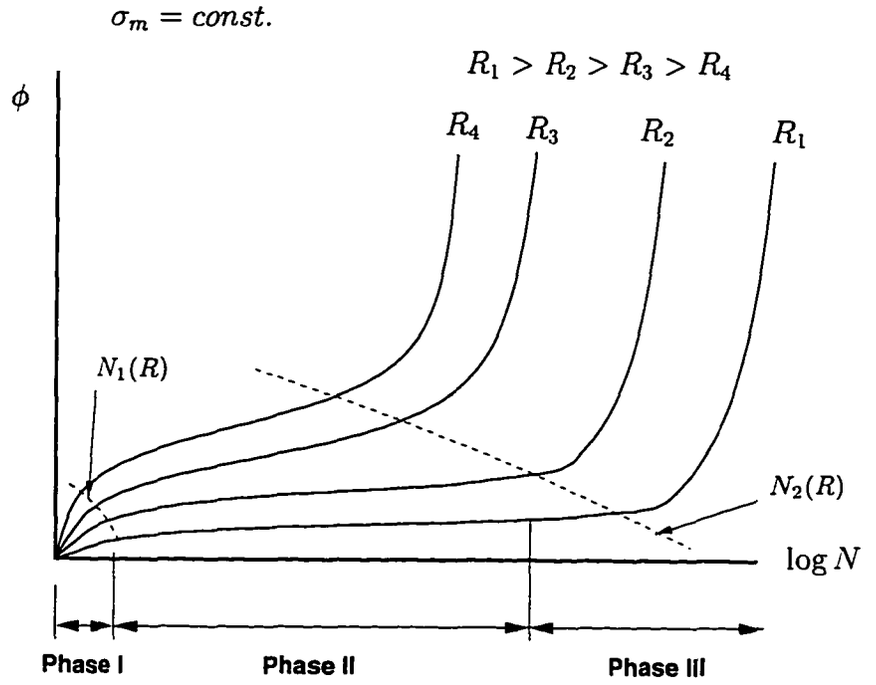

Figure 4.10: Fatigue damage evolution curves ($\phi - N$ curves) for constant σ_{mean}

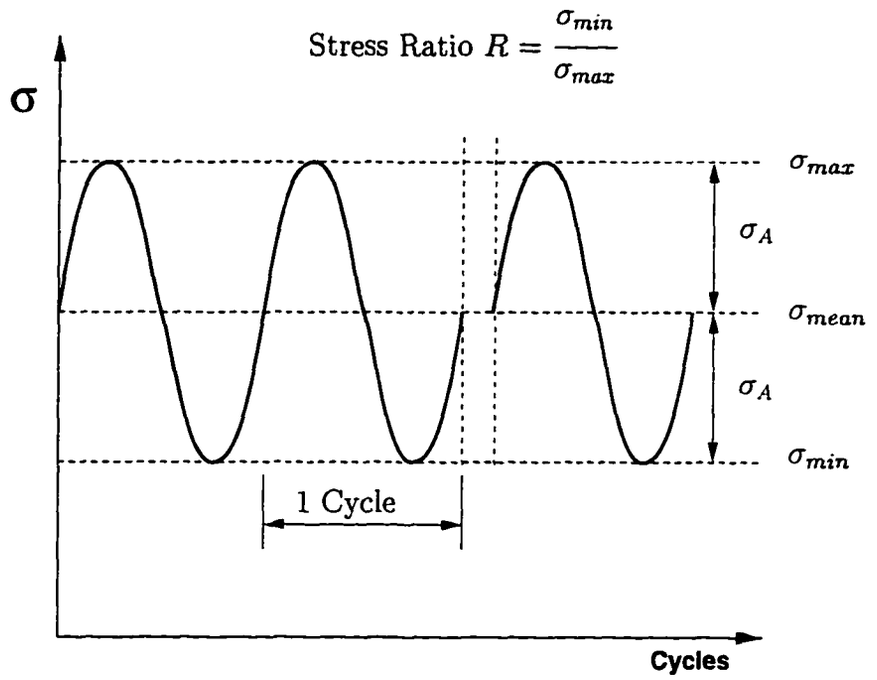

Figure 4.11: Definition of a fatigue cycle with appropriate variables

Another common quantity referred to in fatigue analysis is the stress amplitude of the applied loading, σ_A. This quantity is also shown in Figure 4.11, and is defined as

$$\sigma_A = \frac{1}{2}(\sigma_{max} - \sigma_{min}) \tag{4.19}$$

To obtain a relation for ξ over a range of mean stresses with constant stress ratio R, or over a range of stress ratios R with a constant mean stress σ_{mean}, one needs to establish the evolution equations for ξ. These equations should cover the entire range that one is interested in. Upon use of similar expressions for the evolution equations for ξ for the curves, one may establish a relationship for the region boundaries N_1 and N_2 (Figures 4.9 and 4.10), with respect to the appropriate parameters σ_{mean} and R. Subsequently it is possible to establish *Wöhler curves* (e.g. Figure 4.12) as well as fatigue damage evolution curves (e.g. Figure 4.7) for any pair of stress ratio and mean stress within the covered range. Fatigue damage evolution curves are especially

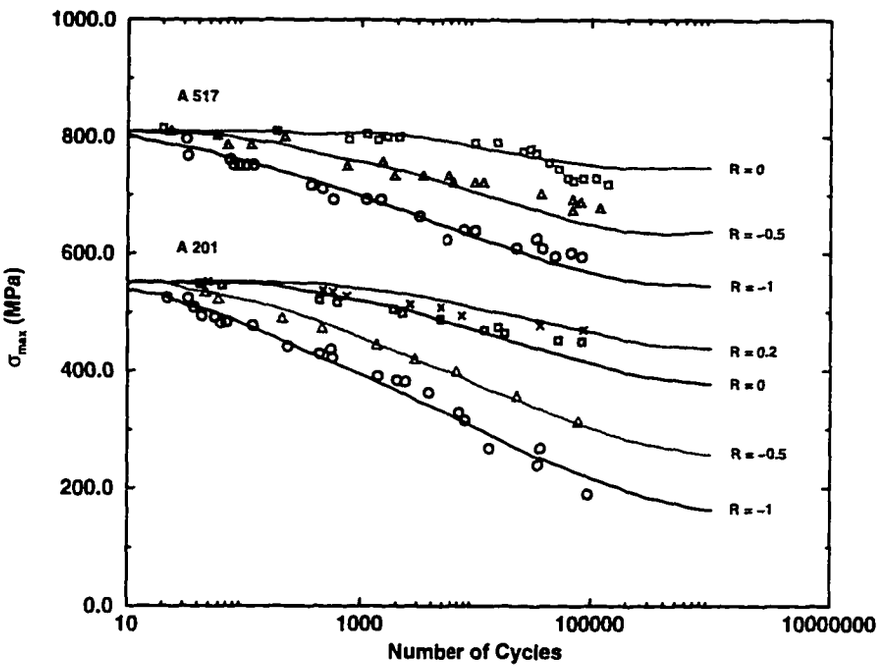

Figure 4.12: Example of *Wöhler* curves for $A\,201$ and $A\,517$ steels at room temperature (Lemaitre and Chaboche, 1990)

of interest, since they allow for residual strength analysis of the composite material at any point in time during its fatigue life time. In general *Wöhler curves* represent the accumulation of experimentally obtained fatigue failure data, where the number of cycles to failure, N_f, is plotted versus the applied maximum stress σ_{max} for a fixed stress ratio R. *Wöhler curves* where the number of cycles to failure N_f are plotted versus the applied mean stress, σ_{mean}, maybe found occasionally in the literature due to the prohibitive cost in determining them experimentally.

4.2 Numerical Implementation of the Model

The numerical implementation of the proposed micro-mechanical fatigue damage model is performed in full tensor notation. This is, none of the tensors used in the theoretical derivation are reduced to a lower order than its original order. This is mainly done to allow for simpler but more robust implementation, and furthermore, the code development is made easier with respect to debugging, while it is understood that the efficiency of the implementation maybe increased. However, it should be emphasized again that robustness of the code has the highest priority over the efficiency due to the availability of high performance computational equipment. Improvement in the overall loading algorithm is implemented through an adaptive stepsize method which shortens the execution time of the initial implementation by as much as 400%. Full tensor implementation is also used to avoid laborious checking of symmetry properties of tensors during the transformation from higher to lower order tensors, which is for the cases of sixth and eight order tensors extremely cumbersome. However, it is necessary to convert the eight order tensors, arising from the second order partial derivatives of the damage effect tensor M_{ijkl} with respect to the damage variable ϕ_{mn} and ϕ_{pq} (see equation 3.29), to a vector representation due to limitations in the compiler available. It allows one to use multi-dimensional arrays of order 7 or lower only.

4.2.1 Micro-Mechanical Implementation and Loading

As outlined before in Section 3.3, the proposed model uses a micro-mechanical approach through the use of stress concentration tensors based on the Mori-Tanaka method (Mori and Tanaka, 1973; Benveniste, 1987). In this approach the external applied loading is distributed to the individual constituents based on principles

involving kinetic and kinematic boundary conditions (refer to Section 3.3.1). The material behavior of the constituents is then established individually, before a homogenization technique is employed to obtain the overall composite material behavior (refer to Section 3.5).

The applied loading is given in the form of a sinusoidal wave as (see Figure 4.11)

$$\sigma_{ij} = \sigma_{ij,mean} + \sigma_{ij,A} \sin\theta \tag{4.20}$$

where

$$\begin{aligned} \sigma_{ij,mean} &= \text{applied mean stress} \\ \sigma_{ij,A} &= \text{applied stress amplitude} \\ \theta &= \text{phase angle in radians} \end{aligned}$$

For the numerical integration scheme an adaptive stepsize algorithm is implemented such that the stress increments are chosen based on few numerical analyses over a few fatigue cycles in order to check numerical sensitivity of the obtained results based on the adopted stepsize. From this study the following stepsize is used

$$\begin{aligned} \Delta\sigma_{ij} &= \frac{\sigma_{ij,mean}}{25} \quad \text{if } \sigma_{ij} < \sigma_{mean,ij} \quad \text{(non-damage state)} \\ \Delta\sigma_{ij} &= 1\,MPa \quad \text{if } \sigma_{ij} < \sigma_{mean,ij} \quad \text{(damage state)} \end{aligned} \tag{4.21}$$

during the loading phase to the mean stress and

$$\Delta\sigma_{ij} = (\sin(\theta + \Delta\theta) - \sin\theta) \times \sigma_{ij,A} \quad \text{(during cyclic loading)} \tag{4.22}$$

with

$$\begin{aligned} \Delta\theta &= \frac{\pi}{50} \quad \text{(during a non-damaging state)} \\ \Delta\theta &= \frac{\pi}{900} \quad \text{(during a damaging state)} \end{aligned}$$

for the cyclic loading phase. The stepsize is maintained at the minimum values outlined above during damage in either constituents, and is increased back to the maximum stepsize as soon as damage diminishes. This occurs on the descending branch of the sinusoidal wave.

In the following the implemented procedure to investigate damage evolution in the individual constituents is outlined, where reference is made to the flowchart shown in Figure 4.13. After calculating the stress increment for the next load step the constituent stresses and stress increments are calculated based on the Mori-Tanaka stress concentration tensors as given in Section 3.3.1.1. Each constituent is then considered individually and investigated if the current applied stress increment induces further damage. This is done using the conditions for damage evolution as specified in equation (3.98). If the conditions for damage evolution are satisfied, the appropriate damage increment based on the current stress increment in the constituent is calculated using equation (3.103). The next step requires to update the appropriate variables affected by a change in the damage variable ϕ. After updating the newly calculated values for the damage variable ϕ they are used to verify that the new stress and damage state satisfies the damage surface with a margin of error of $|g| \leq 10^{-3}$. In the case that the criterion is not satisfied a return algorithm as described in Section 4.2.2 is employed to return the image stress point to the damage surface. If this error criterion is satisfied the procedure is continued for the next constituent following the same procedure as outlined in order to obtain the overall composite material damage. Subsequently the overall damage effect tensor is calculated together with the overall composite material behavior as outlined in Section 3.5. The complete procedure is repeated for each load increment and over all loading cycles.

It should be emphasized that in the event of damage, the quantities needed to calculate the new effective stress and strain concentration tensors (equations 3.41

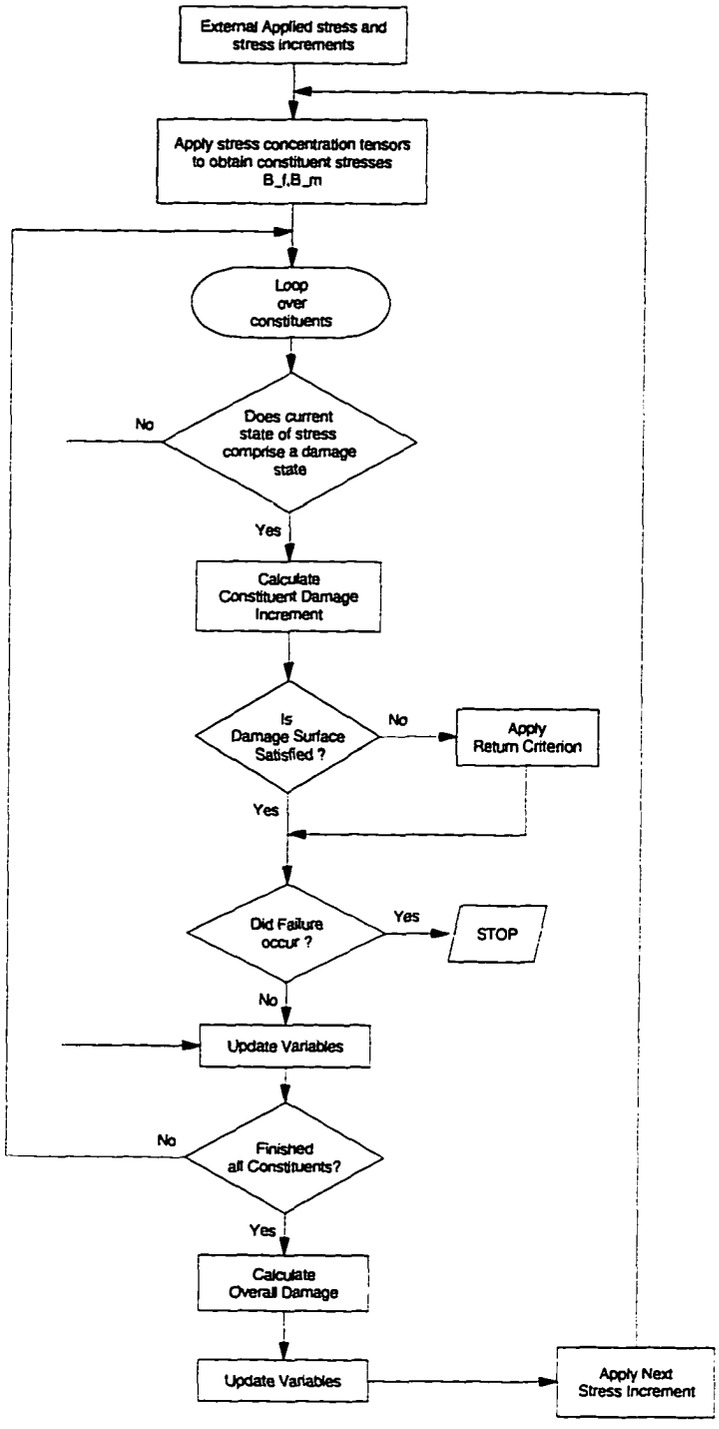

Figure 4.13: Flowchart showing the core of the micro-mechanical fatigue damage model

and 3.55) as well as the effective volume fractions (equations 3.56) will be determined after each increment. The change in the effective volume fractions during the fatigue damage evolution is shown in Figure 4.14 while the appropriate change

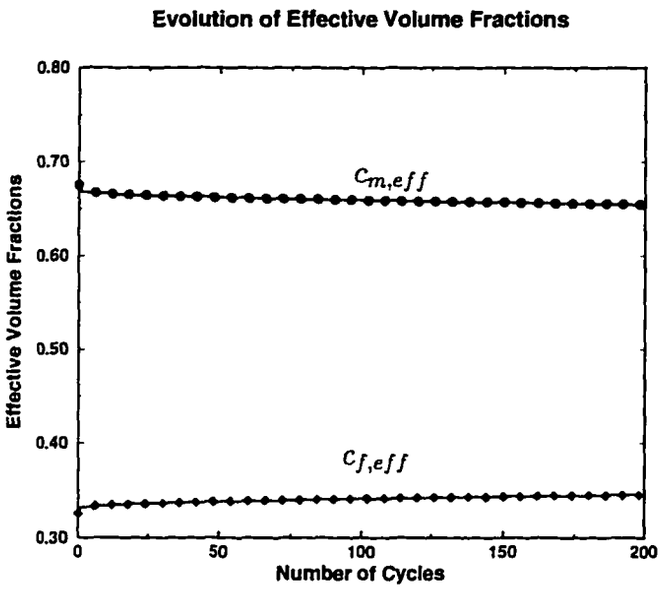

Figure 4.14: Evolution of effective volume fractions with fatigue damage evolution

in the effective stress during the fatigue damage evolution is shown in Figure 4.15. Both graphs are obtained for the case of a uni-axial fatigue loading such as those used in the simulations described in Chapter 5. The appropriate evolution of the damage in the fiber and the matrix is shown in Figure 4.16. Failure in the material is defined following the recommendation of Lemaitre (1985b) and Lemaitre and Chaboche (1990) by using a critical damage value of $\phi_{crit} = 0.4$ for all components. According to Lemaitre (1985b) and Lemaitre and Chaboche (1990), critical values maybe used for different materials.

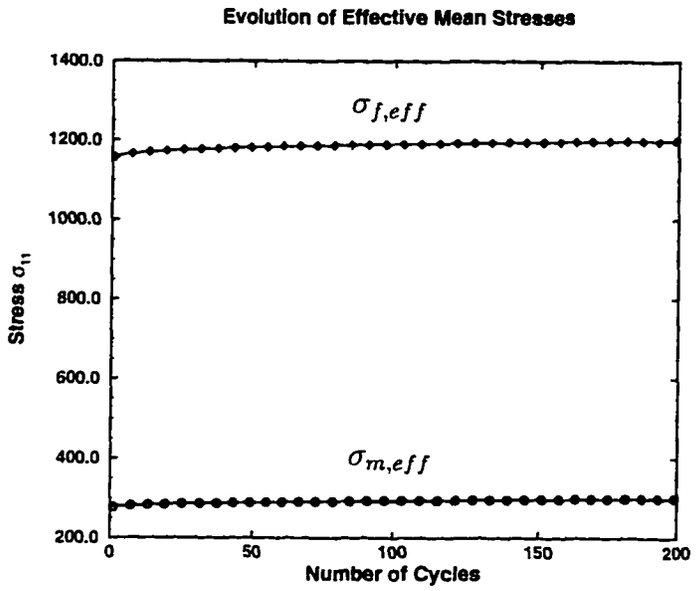

Figure 4.15: Evolution of effective stresses in the constituents with fatigue damage evolution

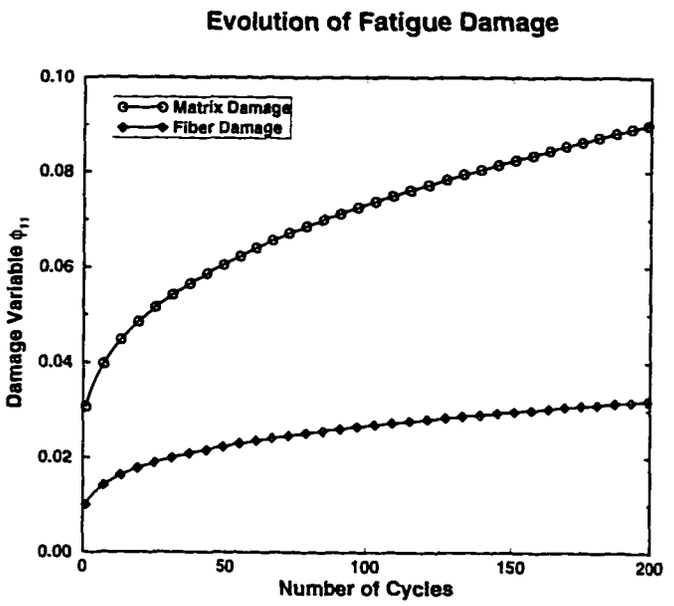

Figure 4.16: Fatigue damage evolution in the constituents

4.2.2 Return to the Damage Surface

In the numerical implementation of the model, it appears that after calculating the damage increment $d\boldsymbol{\phi}$ for the current stress increment $d\boldsymbol{\sigma}$ and updating all the appropriate parameters depending on the damage variable $\boldsymbol{\phi}$, the damage condition is in general greater than zero. Therefore it is necessary to return the new image point to the damage surface by employing an appropriate return criterion.

At the end of the n^{th} increment one assumes that the damage surface g is satisfied such that

$$g^{(n)}\left(\boldsymbol{\sigma}^{(n)}, \boldsymbol{\phi}^{(n)}, \kappa^{(n)}, \boldsymbol{\gamma}^{(n)}\right) = 0 \tag{4.23}$$

Assuming a damage loading process the next stress increment $d\boldsymbol{\sigma}$ (assuming a damage loading) will result in a damage increment $d\boldsymbol{\phi}$ which is obtained using equation (3.103). After calculating the damage increment, all the other variables affected by a change in the damage variable, such as κ and $\boldsymbol{\gamma}$, will be updated. Subsequent checking of the damage surface (equation 3.87) with the updated values for $\boldsymbol{\sigma}$, $\boldsymbol{\phi}$, κ and $\boldsymbol{\gamma}$ will generally yield the following condition as

$$g^{(n+1)}\left(\boldsymbol{\sigma}^{(n+1)}, \boldsymbol{\phi}^{(n+1)}, \kappa^{(n+1)}, \boldsymbol{\gamma}^{(n+1)}\right) > 0 \tag{4.24}$$

where

$$\boldsymbol{\sigma}^{(n+1)} = \boldsymbol{\sigma}^{(n)} + d\boldsymbol{\sigma}^{(n+1)} \tag{4.25}$$

$$\boldsymbol{\phi}^{(n+1)} = \boldsymbol{\phi}^{(n)} + d\boldsymbol{\phi}^{(n+1)} \tag{4.26}$$

$$\kappa^{(n+1)} = \kappa^{(n)} + d\kappa^{(n+1)} \tag{4.27}$$

$$\boldsymbol{\gamma}^{(n+1)} = \boldsymbol{\gamma}^{(n)} + d\boldsymbol{\gamma}^{(n+1)} \tag{4.28}$$

Upon expansion of the left hand side of equation (4.24) using a *Taylor series* expansion of order one, the following expression is obtained as

$$g^{(n+1)}\left(\boldsymbol{\sigma}^{(n)}+d\boldsymbol{\sigma}^{(n+1)},\boldsymbol{\phi}^{(n)}+d\boldsymbol{\phi}^{(n+1)},\kappa^{(n)}+d\kappa^{(n+1)},\boldsymbol{\gamma}^{(n)}+d\boldsymbol{\gamma}^{(n+1)}\right)$$
$$=g^{(n)}\left(\boldsymbol{\sigma}^{(n)},\boldsymbol{\phi}^{(n)},\kappa^{(n)}\boldsymbol{\gamma}^{(n)}\right)+\left.\frac{\partial g}{\partial\boldsymbol{\sigma}}\right|^{(n)}d\boldsymbol{\sigma}^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\phi}}\right|^{(n)}d\boldsymbol{\phi}^{(n+1)}$$
$$+\left.\frac{\partial g}{\partial\kappa}\right|^{(n)}d\kappa^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\gamma}}\right|^{(n)}d\boldsymbol{\gamma}^{(n+1)}>0 \tag{4.29}$$

Making use of equations (3.89) and (3.92) and substituting them into equation (4.29) one obtains

$$g^{(n+1)}\left(\boldsymbol{\sigma}^{(n)}+d\boldsymbol{\sigma}^{(n+1)},\boldsymbol{\phi}^{(n)}+d\boldsymbol{\phi}^{(n+1)},\kappa^{(n)}+\boldsymbol{Y}^{(n)}:d\boldsymbol{\phi}^{(n+1)},\boldsymbol{\gamma}^{(n)}+\mathbf{c}\,d\boldsymbol{\phi}^{(n+1)}\right)$$
$$=g^{(n)}\left(\boldsymbol{\sigma}^{(n)},\boldsymbol{\phi}^{(n)},\kappa^{(n)}\boldsymbol{\gamma}^{(n)}\right)+\left.\frac{\partial g}{\partial\boldsymbol{\sigma}}\right|^{(n)}d\boldsymbol{\sigma}^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\phi}}\right|^{(n)}d\boldsymbol{\phi}^{(n+1)}$$
$$+\left.\frac{\partial g}{\partial\kappa}\right|^{(n)}\boldsymbol{Y}^{(n+1)}:d\boldsymbol{\phi}^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\gamma}}\right|^{(n)}\mathbf{c}\,d\boldsymbol{\phi}^{(n+1)}>0 \tag{4.30}$$

In order to obtain an equality in equation (4.30), and hence satisfying the damage surface with $g^{(n+1)}=0$, one introduces a linear coefficient α to adjust the damage increment $\boldsymbol{\phi}$ such that one may express equation (4.30) as

$$g^{(n+1)}\left(\boldsymbol{\sigma}^{(n)}+d\boldsymbol{\sigma}^{(n+1)},\boldsymbol{\phi}^{(n)}+\alpha\,d\boldsymbol{\phi}^{(n+1)},\kappa^{(n)}+\alpha\boldsymbol{Y}^{(n)}:d\boldsymbol{\phi}^{(n+1)},\right.$$
$$\left.\boldsymbol{\gamma}^{(n)}+\alpha\,\mathbf{c}\,d\boldsymbol{\phi}^{(n+1)}\right)$$
$$=g^{(n)}\left(\boldsymbol{\sigma}^{(n)},\boldsymbol{\phi}^{(n)},\kappa^{(n)}\boldsymbol{\gamma}^{(n)}\right)+\left.\frac{\partial g}{\partial\boldsymbol{\sigma}}\right|^{(n)}d\boldsymbol{\sigma}^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\phi}}\right|^{(n)}\alpha\,d\boldsymbol{\phi}^{(n+1)}$$
$$+\left.\frac{\partial g}{\partial\kappa}\right|^{(n)}\alpha\boldsymbol{Y}^{(n+1)}:d\boldsymbol{\phi}^{(n+1)}+\left.\frac{\partial g}{\partial\boldsymbol{\gamma}}\right|^{(n)}\mathbf{c}\,\alpha\,d\boldsymbol{\phi}^{(n+1)}=0 \tag{4.31}$$

Solving equation (4.31) for the coefficient α yields

$$\alpha=-\frac{\left(g^{(n)}+\left.\frac{\partial g}{\partial\boldsymbol{\sigma}}\right|^{(n)}d\boldsymbol{\sigma}^{(n+1)}\right)}{\left(\left.\frac{\partial g}{\partial\boldsymbol{\phi}}\right|^{(n)}+\left.\frac{\partial g}{\partial\kappa}\right|^{(n)}\boldsymbol{Y}^{(n)}+\mathbf{c}\left.\frac{\partial g}{\partial\boldsymbol{\gamma}}\right|^{(n)}\right)d\boldsymbol{\phi}^{(n+1)}} \tag{4.32}$$

or with

$$\frac{\partial g}{\partial\boldsymbol{\gamma}}=-\frac{\partial g}{\partial\boldsymbol{Y}} \tag{4.33}$$

one obtains

$$\alpha = -\frac{\left(g^{(n)} + \frac{\partial g}{\partial \sigma}\Big|^{(n)} d\sigma^{(n+1)}\right)}{\left(\frac{\partial g}{\partial \phi}\Big|^{(n)} + \frac{\partial g}{\partial \kappa}\Big|^{(n)} Y^{(n)} - c\,\frac{\partial g}{\partial Y}\Big|^{(n)}\right) d\phi^{(n+1)}} \tag{4.34}$$

The employment of the above procedure to satisfy the damage surface at any instant of damage evolution proved satisfactory throughout the analyses except at the very first occurrence of damage which is shown in Figures 4.17 and 4.18. Figure 4.17

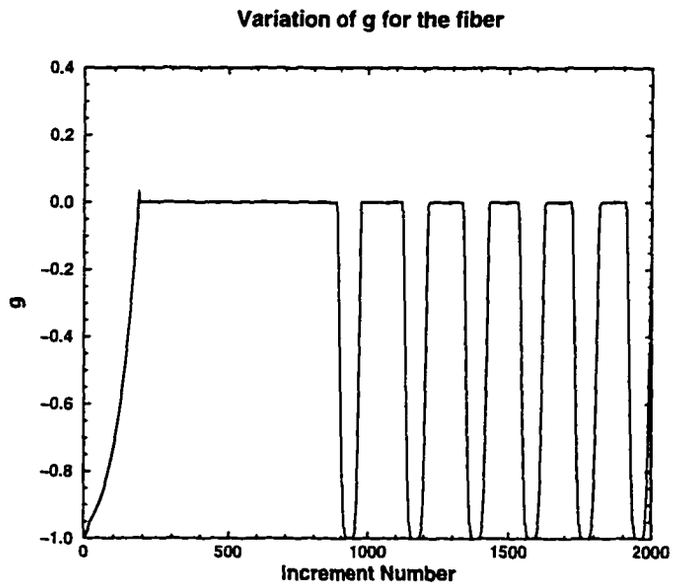

Figure 4.17: Evolution of g for the fiber

shows the evolution of g for the case of the fiber, while Figure 4.18 shows the same information for the case of the matrix over a number of cycles. It is clear from this figures that the above mentioned inconsistency in satisfying the damage surface appropriately, occurs only at the very first instant of damage while at any later point in time the damage surface is satisfied appropriately. The inconsistency is attributed to the point of discontinuity in the damage evolution. Through several analyses it is found that the error at this point with respect to g maybe reduced by further

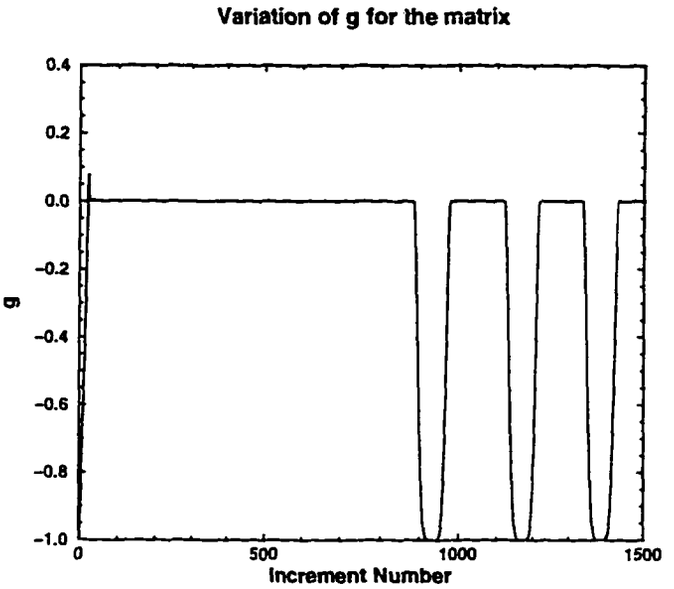

Figure 4.18: Evolution of g for the matrix

reducing the stepsize. The error in the current implementation is $\delta \leq 0.08$ for the initial onset of damage for the case of the matrix and $\delta \leq 0.03$ for the case of the fiber. At any later occurrence of damage the error is reduced to below $\delta \leq 0.001$. This margin of error for the initial onset of damage is acceptable since it is also found that it has no influence on the magnitude of the subsequent damage evolution.

4.3 Summary

The micro-mechanical damage theory for high cycle fatigue damage in metal matrix composites as described in Chapter 3 is implemented into a numerical analysis code. The implementation is performed using full tensor notation in order to simplify debugging as well as implementation, and to avoid the complications during the reduction in the order of the employed tensors. The numerical model is used to study the parameters employed in the micro-mechanical model, and to show the influence of these parameters on the damage initiation and evolution process during the fatigue

life of the material. Furthermore the procedures to obtain the necessary material and model parameters are described. The main part of the numerical model is shown and explained by means of a flowchart. Numerical problems occuring during the numerical simulations are described and appropriate methods to reduce the numerical errors or eliminate them are described.

In the following the numerical implementation of the micro-mechancial fatigue damage model is employed to simulate high cycle fatigue damage at room temperature in a uni-directional continuous fiber reinforced metal matrix composite material system, namely *SCS-6/Ti-15-3*. Results of the numerical simulations are presented in the form of graphs showing the fatigue damage evolution in the individual constituents, the change in effective volume fractions and the change in effective mean stresses in the constituents. Furthermore the model is validated by comparing the final results for the number of cycles to failure obtained for several different loading cases with those obtained from the literature for this specific material system.

Chapter 5
Numerical Simulations

In the following the numerical implementation of the micro-mechanical fatigue damage model as described in Section 4.2 is employed to simulate fatigue tests on uni-directional metal matrix composites. As mentioned previously the current model is developed to model and simulate only high cycle fatigue loading which covers in general ranges of fatigue loading of approximately 50,000 cycles and more. High cycle fatigue loading is characterized by only elastic deformations at the macro scale, while micro-hardening is present in the material as described in Section 3.4.4.

5.1 Material Properties and Model Parameters

The material considered for the fatigue simulations is a uni-directional continuous fiber reinforced metal matrix composite denoted by *SCS-6/Ti-15-3*. Here the fiber identification *SCS-6* reflects the manufacturer of the fiber while the matrix identification *Ti-15-3* provides for the chemical composition of the matrix alloy. In the current case the matrix is made of titanium *Ti* as the base material with several alloy compounds, such as 15 *Vol.-%* Vanadium *V*, 3 *Vol.-%* Chromium *Cr*, 3 *Vol.-%* Aluminum *Al*, and 3 *Vol.-%* Tin *Sn* (Jeng et al., 1991a). This material system is chosen due to the availability of some experimental data for a uni-directionally continuous fiber reinforced metal matrix composite subjected to fatigue loading at room temperature. While a large amount of experimental data are available for general lay-ups, such as $[(0/90)]_{2s}$, $[0/\pm 45/90]_s$ and $[(0)_2/\pm 45]_s$, only very little data could

be located for high-cycle fatigue of uni-directional continuous fiber reinforced metal matrix composites at room temperature. The material properties for the individual constituents are obtained from the literature, namely from Johnson (1989), and are given in Table 5.1. As mentioned in Section 4.1 it is common to consider the

Table 5.1: Material properties used in the analysis

	$E\,(GPa)$	ν	$\sigma_u\,(MPa)$	$\sigma_y\,(MPa)$	c (in %)
Matrix *(Ti-15-3)*	91.8	0.36	933.6	689.5	67.5
Fiber *(SCS-6)*	400.0	0.25	N/A	N/A	32.5

constituents as isotropic materials, which is followed in the numerical simulations discussed in this chapter also. Hence, the material properties given in Table 5.1 represent the isotropic properties of the constituents used. The composite is assumed to be made of continuous Silicon Carbide fibers (*SCS-6*) occupying a volume fraction of 32.5% and a Titanium alloy matrix compound of the alloy materials as given before.

The model parameters used in the simulations are derived or obtained as outlined in Section 4.1. The set of fixed properties and parameters used in the simulations is given in Table 5.2, where reference is made to the model parameters which need

Table 5.2: Model parameters used in the simulations

	$V\,(MPa)$	$\lambda\,(MPa)$	ξ
Matrix *(Ti-15-3)*	0.05	86786	refer to Eqs. (5.1) - (5.3)
Fiber *(SCS-6)*	1.0	160000	refer to Eqs. (5.1) - (5.3)

to be adjusted for a particular simulation, such as the parameter ξ. As explained in Section 4.1.2, the parameter ξ is dependent on the applied loading parameters

and is expressed as a function of the number of applied loading cycles N. Due to the lack of appropriate experimental results and data, the following expressions for ξ are assumed for the present simulation based on the experimental observations made during fatigue experiments as stated by Dieter (1988) and Suresh (1991) which is reflected in the qualitative $\phi - N$ curve as shown in Figure 4.6, as well as based on the trends of experimental results depicted in Figures 4.7 and 4.8. As stated before (Section 4.1.2), the initial damage phase (Phase I) covers in general only a few cycles, hence a very small number of cycles is used as an estimation for N_1 together with a parabolic approximation representing a fairly high initial damage rate. Phase II, which covers a major part of the fatigue life of a material (Dieter, 1988; Suresh, 1991), is characterized by a fairly constant and steady damage initiation and propagation, reflected by the relative large value chosen for the parameter N_2 (Table 5.3) and the approximation in the form of an exponential function. Phase III, representing the propagation and failure phase of the fatigue life, is approximated also by a parabolic function where the highly non-linear behavior in the damage evolution (Dieter, 1988; Suresh, 1991), exhibited in the material prior to failure (e.g. Figures 4.7 and 4.8), maybe adjusted through the coefficient a_3 (refer to equations 5.3) to fit the experimental data. The appropriate expressions for the individual phases are given as

Phase I:

$$\xi_I(\mathcal{N}) = a_1 \mathcal{N}^2 + b_1 \mathcal{N} + c_1 \tag{5.1a}$$

where

$$a_1 = \frac{\xi_1}{\mathcal{N}_1} \frac{(\ln \xi_2 - \ln \xi_1)}{(\mathcal{N}_2 - \mathcal{N}_1)} - \frac{(\xi_1 - \xi_0)}{\mathcal{N}_1^2} \tag{5.1b}$$

$$b_1 = 2 \frac{(\xi_1 - \xi_0)}{\mathcal{N}_1} - \xi_1 \frac{(\ln \xi_2 - \ln \xi_1)}{(\mathcal{N}_2 - \mathcal{N}_1)} \tag{5.1c}$$

$$c_1 = \xi_0 \tag{5.1d}$$

Phase II:

$$\xi_{II}(\mathcal{N}) = a_2\, e^{b_2 (\mathcal{N}-\mathcal{N}_1)} \tag{5.2a}$$

where

$$a_2 = \xi_2 \frac{(\ln \xi_2 - \ln \xi_1)}{(\mathcal{N}_2 - \mathcal{N}_1)} \tag{5.2b}$$

$$b_2 = \frac{(\ln \xi_2 - \ln \xi_1)}{(\mathcal{N}_2 - \mathcal{N}_1)} \tag{5.2c}$$

Phase III:

$$\xi_{III}(\mathcal{N}) = a_3\, (\mathcal{N} - \mathcal{N}_2)^2 + b_3\, (\mathcal{N} - \mathcal{N}_2) + c_3 \tag{5.3a}$$

where

$$a_3 = \text{(based on experimental data)} \tag{5.3b}$$

$$b_3 = \xi_2 \frac{(\ln \xi_2 - \ln \xi_1)}{(\mathcal{N}_2 - \mathcal{N}_1)} \tag{5.3c}$$

$$c_3 = \xi_2 \tag{5.3d}$$

In all the above equations the parameters $\mathcal{N}$, $\mathcal{N}_1$ and $\mathcal{N}_2$ are defined as

$$\mathcal{N} = \log N$$

$$\mathcal{N}_1 = \log N_1$$

$$\mathcal{N}_2 = \log N_2$$

where N represents the number of cycles, N_1 represents the number of cycles at the boundary between Region I and Region II, and N_2 represents the number of cycles at the boundary between Region II and Region III (refer to Figure 4.6).

The chosen functions for ξ for the individual regions satisfy the following conditions for each individual constituent:

Region I:

$$\xi_I(0) = \xi_0 \tag{5.5a}$$

$$\xi_I(\mathcal{N}_1) = \xi_1 \tag{5.5b}$$

$$\left.\frac{d\xi_I}{d\mathcal{N}}\right|_{\mathcal{N}_1} = \left.\frac{d\xi_{II}}{d\mathcal{N}}\right|_{\mathcal{N}_1} \tag{5.5c}$$

Region II:

$$\xi_{II}(\mathcal{N}_1) = \xi_1 \tag{5.6a}$$

$$\xi_{II}(\mathcal{N}_2) = \xi_2 \tag{5.6b}$$

Region III:

$$\xi_{III}(\mathcal{N}_2) = \xi_2 \tag{5.7a}$$

$$\left.\frac{d\xi_{III}}{d\mathcal{N}}\right|_{\mathcal{N}_2} = \left.\frac{d\xi_{II}}{d\mathcal{N}}\right|_{\mathcal{N}_2} \tag{5.7b}$$

Here the quantities ξ_0, ξ_1 and ξ_2 represent the initial value for ξ, the value for ξ at N_1 cycles and at N_2 cycles, respectively. These values have to be determined from the appropriate experimental data as described in Section 4.1.2.

Due to the lack of such data, the appropriate coefficients for the individual expressions have been estimated through sample analyses with subsequent evaluation of tendencies in damage development. For the material system used in these simulations, it is known that the failure mode is matrix dominated (Jeng et al., 1991a), which has been considered in the estimations. The coefficients are chosen as given in Table 5.3. In the following the results of the numerical simulations using the above mentioned parameters and coefficients are presented.

Table 5.3: Coefficients and Parameters used in the simulations

Simulation	Material	N_1	N_2	ξ_0	ξ_1	ξ_2	a_3	Figures
1	Matrix	10	20000	0.70	0.73	0.79	0.75	5.1,5.7
	Fiber	10	60000	0.65	0.68	0.73	0.1	
2	Matrix	10	30000	0.70	0.73	0.78	0.1	5.2,5.8
	Fiber	10	150000	0.65	0.68	0.69	0.1	
3	Matrix	10	50000	0.70	0.73	0.79	0.5	5.3,5.9
	Fiber	10	100000	0.65	0.68	0.69	0.1	
4	Matrix	10	80000	0.70	0.73	0.78	0.25	5.4,5.10
	Fiber	10	150000	0.65	0.68	0.69	0.1	
5	Matrix	10	200000	0.70	0.73	0.78	0.2375	5.5,5.11
	Fiber	10	400000	0.65	0.68	0.69	0.1	

5.2 Results of the Numerical Fatigue Simulation

Five different numerical fatigue test simulations are conducted using the material properties, the model parameters and the coefficients as given in Tables 5.1, 5.2 and 5.3, respectively. The fatigue loading applied is given in the form of a sinusoidal loading as described by equation (4.20) and the stepsize is chosen according to the provisions made in Section 4.2.1, specifically equations (4.21) and (4.22). For the fatigue loading a uni-axial loading is chosen, where $\sigma_{11} \neq 0$ and $\sigma_{ij} = 0$ for all $i, j \neq 1$, such that the direction of the applied normal stress σ_{11} coincides with the fiber direction of the composite. The magnitudes of the applied mean stresses and the cyclic stress amplitudes for the simulations as well as the stress ratios are given in Table 5.4. The results from the numerical simulations are shown in Figures 5.1 - 5.18.

Table 5.4: Applied cyclic loading used in the high cycle fatigue simulations

Simulation	$\sigma_{11,mean}$	$\sigma_{11,A}$	$R = \frac{\sigma_{11,min}}{\sigma_{11,max}}$	$\sigma_{11,max}$	Cycles to Failure
-	(MPa)	(MPa)	-	(MPa)	-
1	605.0	495.0	0.1	1100	50766
2	577.5	472.5	0.1	1050	70435
3	550.0	450.0	0.1	1000	100044
4	528.0	432.0	0.1	960	131745
5	418.0	342.0	0.1	760	458763

Figures 5.1-5.5 show the damage evolution in the fiber and matrix as well as for the overall composite with respect to the number of applied cycles for the simulations 1, 2, 3, 4 and 5, respectively. All five figures show the qualitative trend for high cycle fatigue loading as described in Section 4.1.2, namely that after a very short initial damage phase the damage evolution is characterized by a fairly constant damage initiation and propagation phase, and finally in the failure phase, the damage evolution shows a highly non-linear behavior. Furthermore the graphs depict a matrix dominated failure mode, which is characteristic for this material system when subjected to high cycle fatigue (Jeng et al., 1991a). Failure in the material is initiated due to matrix failure, where the damage ϕ_{11} in the matrix exceeds the critical value of $\phi_{11,crit} = 0.4$, defined according to the provisions made by Lemaitre (1985b) and Lemaitre and Chaboche (1990). When plotting the results for the fatigue damage evolution in the matrix of the five simulations together (Figure 5.6), a consistent behavior in the damage evolution in the matrix is reflected. It is clearly shown that for

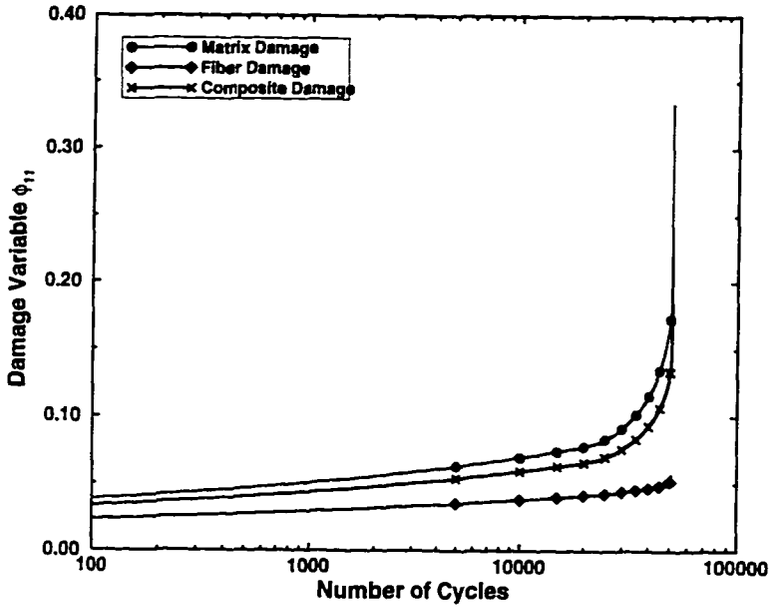

Figure 5.1: Fatigue damage evolution during HCF simulation 1

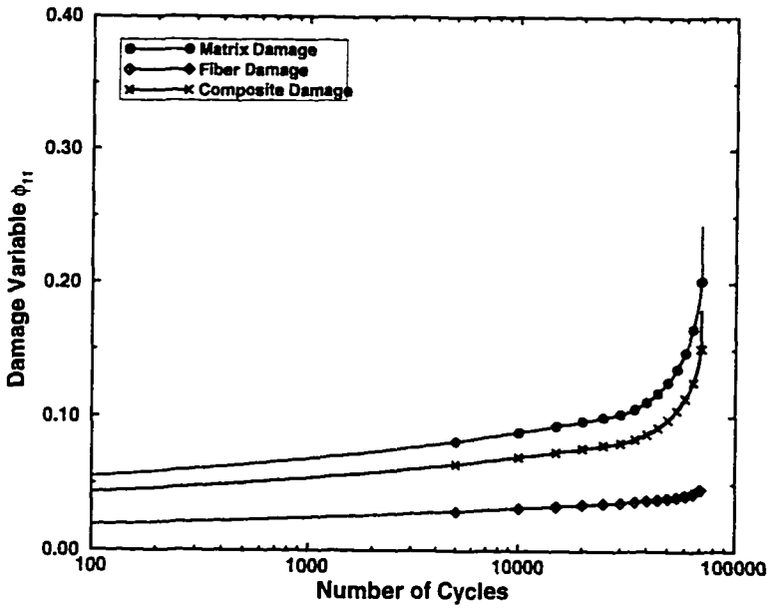

Figure 5.2: Fatigue damage evolution during HCF simulation 2

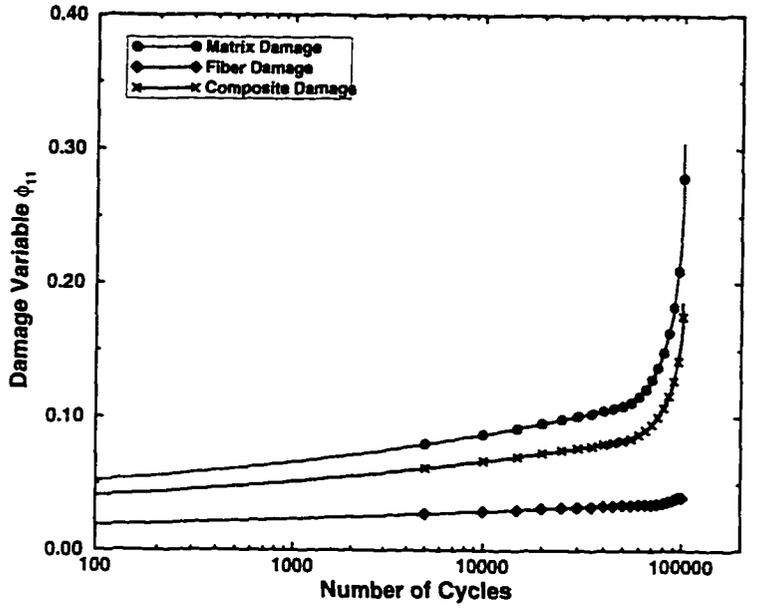

Figure 5.3: Fatigue damage evolution during HCF simulation 3

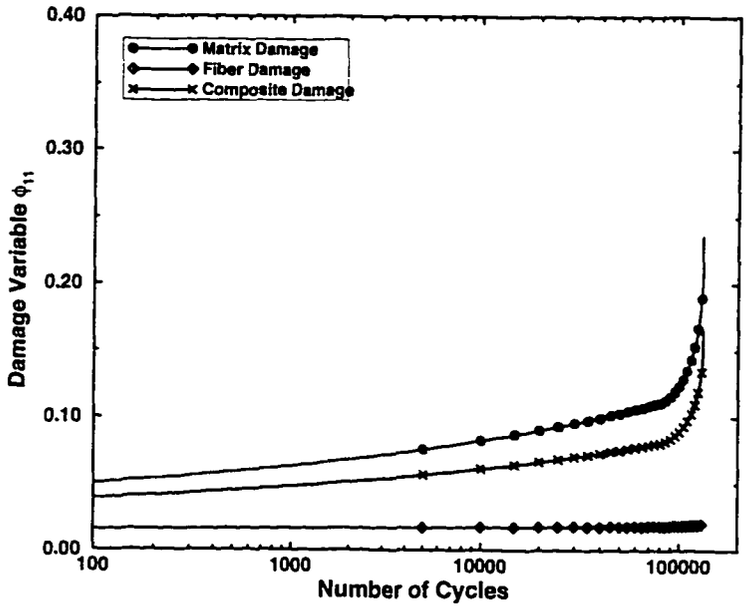

Figure 5.4: Fatigue damage evolution during HCF simulation 4

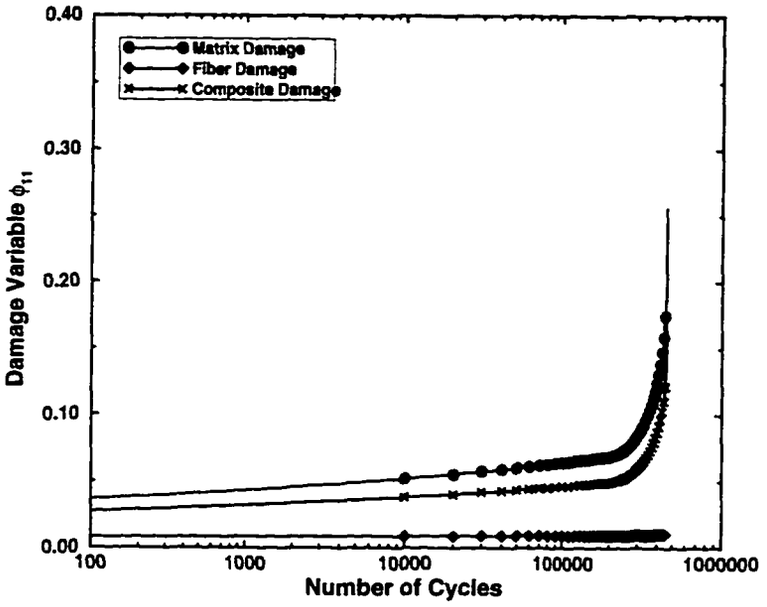

Figure 5.5: Fatigue damage evolution during HCF simulation 5

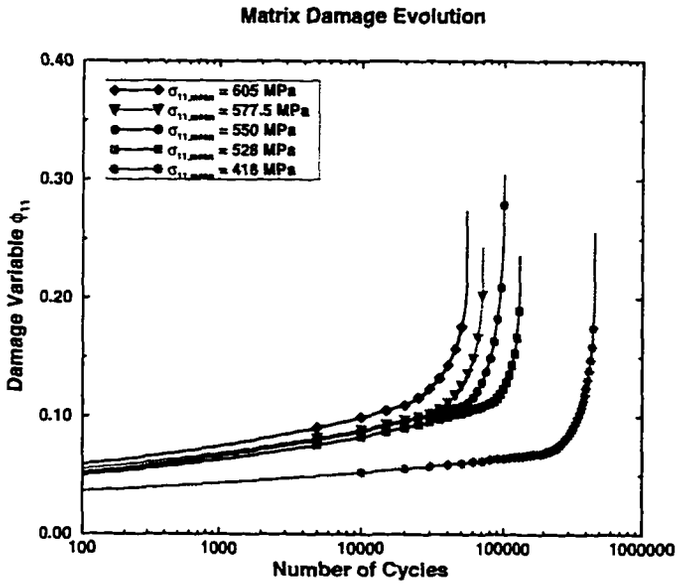

Figure 5.6: Evolution of damage in the matrix

the simulation with the highest mean stress and stress amplitude the matrix damage develops the fastest and failure occurs before failure in any other simulation. Furthermore, all five graphs for the fatigue damage evolution in the matrix show similar behavior, and consistency is depicted for the damage evolution with respect to the magnitudes of the applied mean stresses and the stress amplitudes.

In Figures 5.7 - 5.11 the appropriate changes in the effective volume fractions

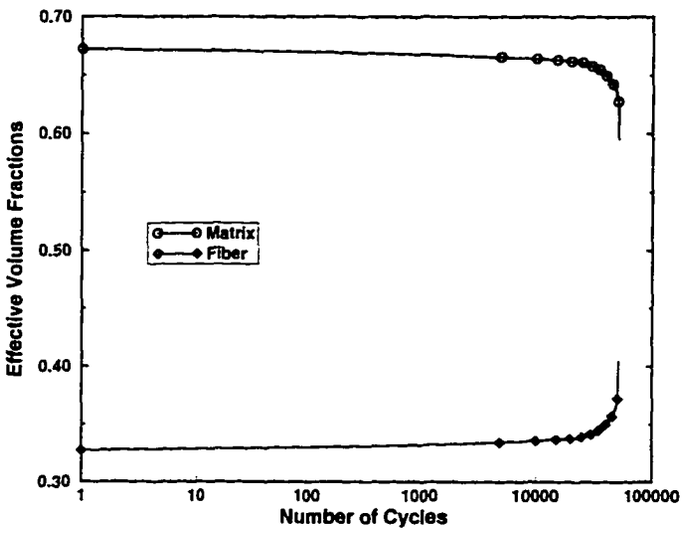

Figure 5.7: Evolution of effective volume fractions (HCF simulation 1)

for the fibers and the matrix for the case of fatigue simulations 1, 2, 3, 4 and 5 are shown. These figures clearly show the change in the effective volume fractions with increasing damage in the constituents. As one may observe, there is an increase in the effective volume fractions for the fibers while the effective volume fractions in the matrix decrease, which is consistent with the behavior shown in Figures 5.1 - 5.5. As expected, the effective volume fraction is increasing for the constituent with the lower damage and damage evolution, in this case the fiber, while the effective volume

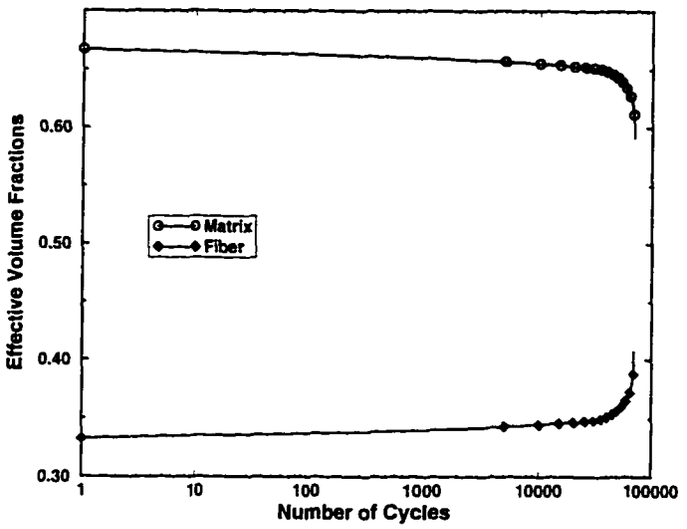

Figure 5.8: Evolution of effective volume fractions (HCF simulation 2)

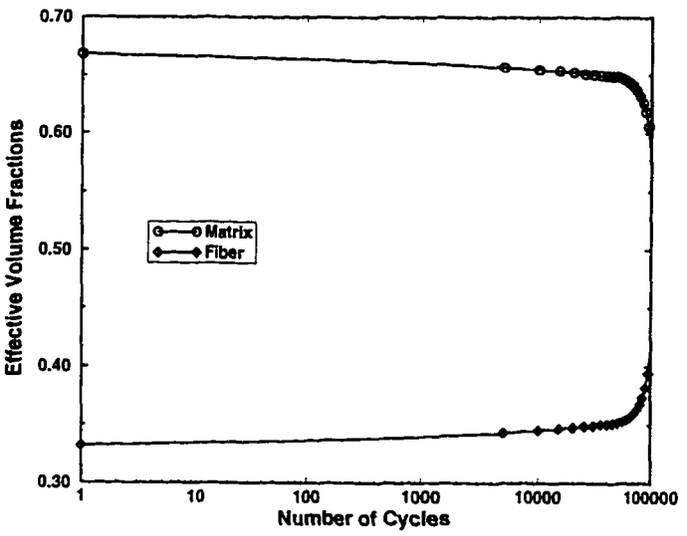

Figure 5.9: Evolution of effective volume fractions (HCF simulation 3)

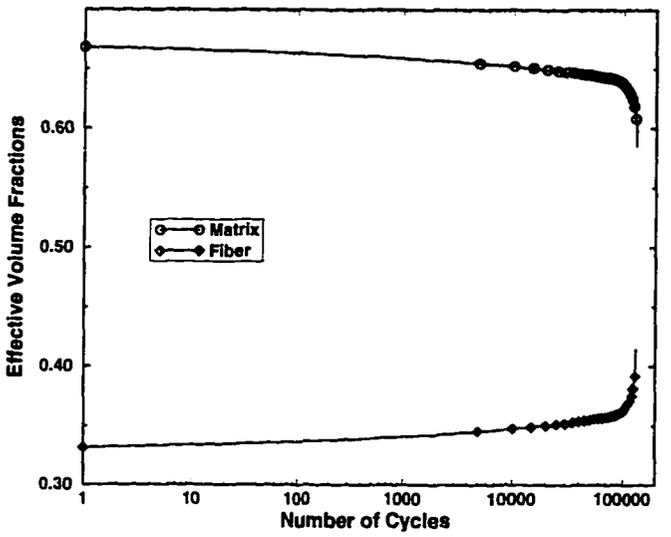

Figure 5.10: Evolution of effective volume fractions (HCF simulation 4)

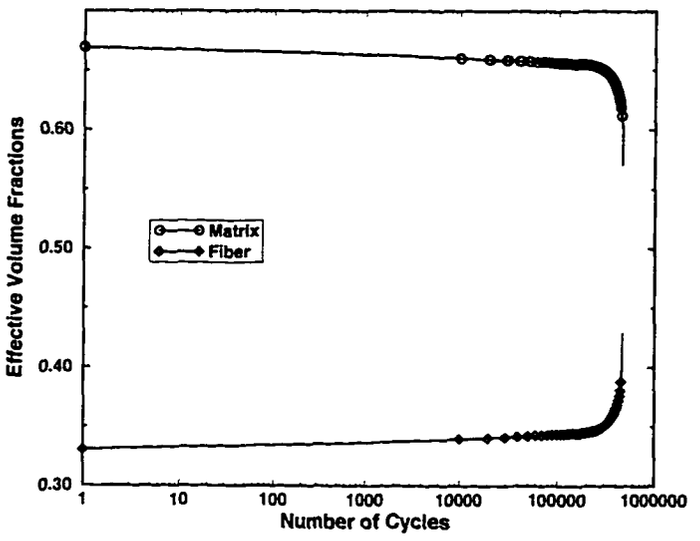

Figure 5.11: Evolution of effective volume fractions (HCF simulation 5)

fraction for the constituent with the higher damage and higher damage evolution, which is in this case the matrix, is decreasing. These results clearly show that the model is able to capture and represent the physical behavior of the damage process in the composite.

A phenomenon associated with the change in effective volume fractions of the constituents is the increase in effective stresses in the constituents. This phenomenon is based on the reduction of effective load resisting area/volume in the composite material which requires the external applied load to be carried by the remaining undamaged material. Since the external applied load is kept constant at a particular load level (load controlled testing) even in the event of damage, an increase in the effective stresses of the composite occurs due to stress redistribution. This change in effective stresses is shown in Figures 5.12 - 5.16 where the variation in the effective

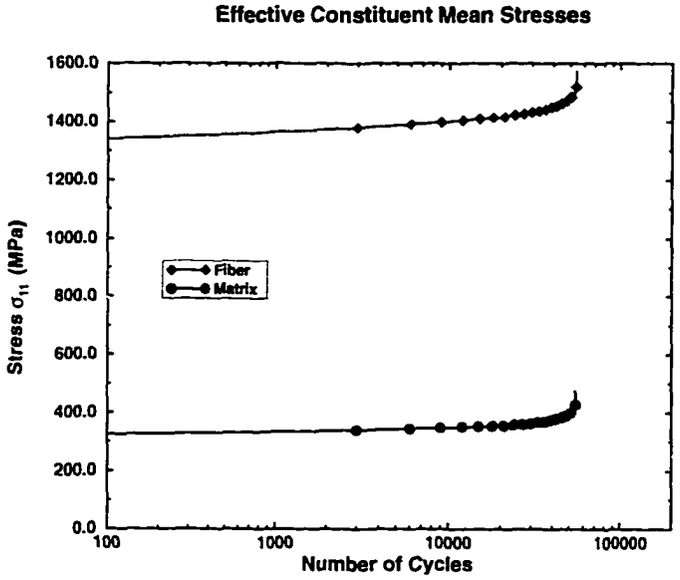

Figure 5.12: Evolution of effective mean stresses (HCF simulation 1)

mean stresses in the constituents is shown with respect to the applied number of

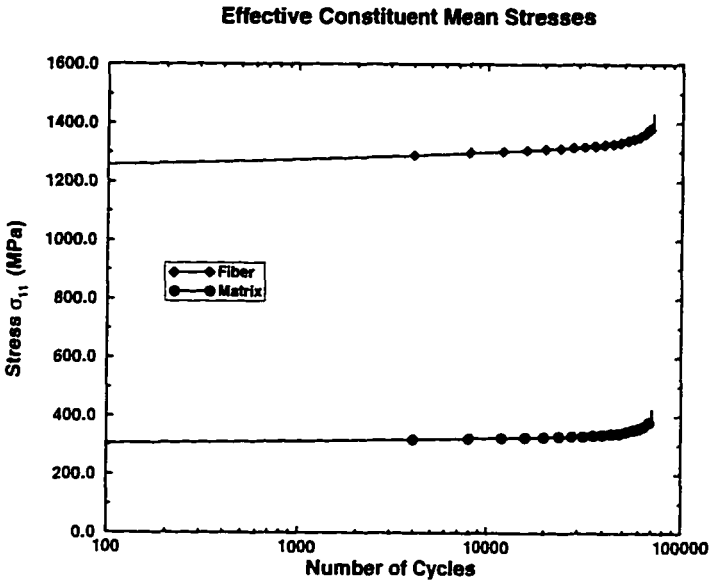

Figure 5.13: Evolution of effective mean stresses (HCF simulation 2)

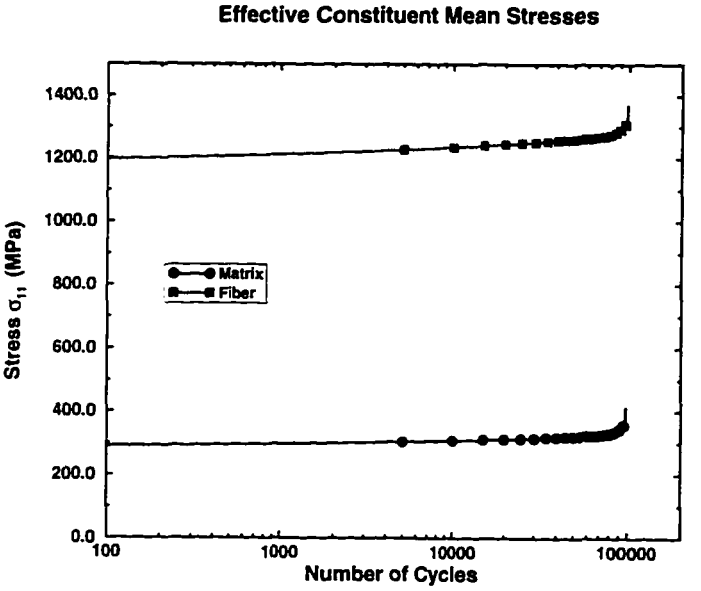

Figure 5.14: Evolution of effective mean stresses (HCF simulation 3)

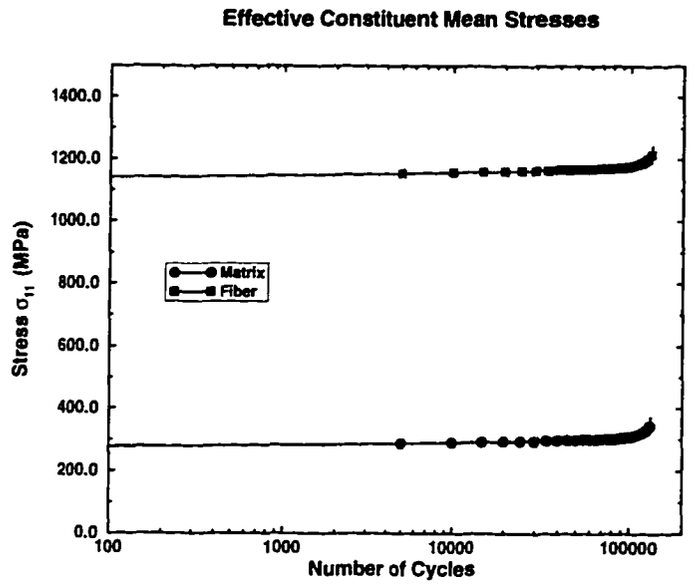

Figure 5.15: Evolution of effective mean stresses (HCF simulation 4)

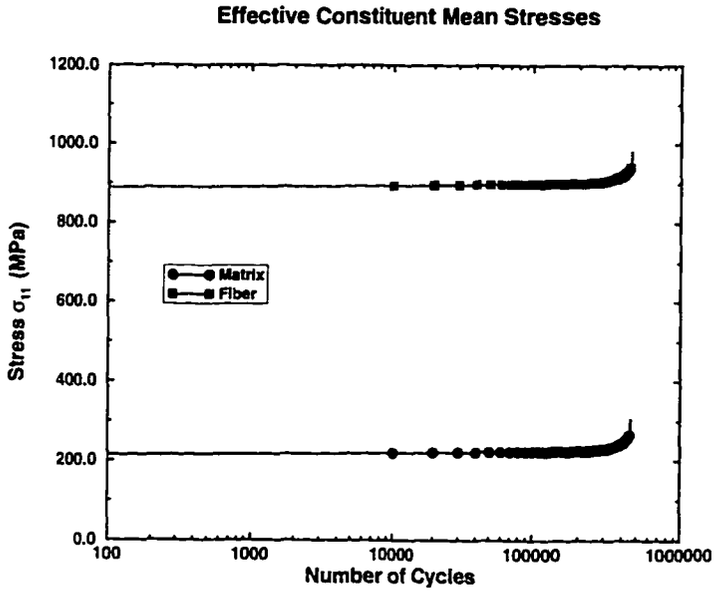

Figure 5.16: Evolution of effective mean stresses (HCF simulation 5)

fatigue cycles. As one may observe, all the graphs clearly show an increase in effective mean stresses with increasing number of fatigue cycles. This is consistent with the behavior of increasing damage in the constituents and hence a reduction in effective load resisting area/volume in the composite material. To show the relative evolution of the effective mean stresses all the effective mean stresses are plotted together as shown in Figure 5.17. The figure clearly shows that the order in magnitude of the

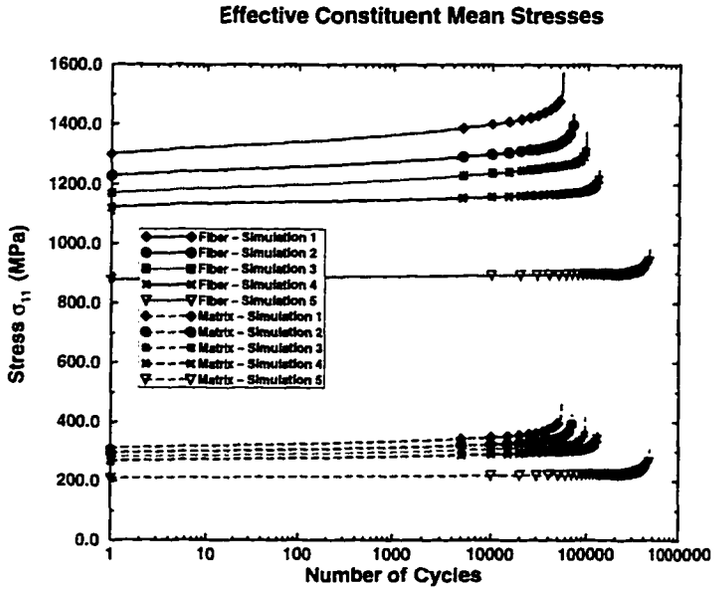

Figure 5.17: Evolution of effective mean stresses for all HCF simulations

applied mean stresses is represented properly throughout the entire simulations.

Furthermore, considering the final number of cycles to failure for the individual tests, a comparison is made with experimental data, which are made available in the form of an *S-N* curve or a *Wöhler diagram* as shown in Figure 5.18. In Figure 5.18 the results of room temperature fatigue experiments on a uni-directional continuous fiber reinforced metal matrix composite are shown as given by Johnson (1989), where the current model results are superimposed. Despite the presence of only a few

experimental data points, it is noted that the model results show a good trend with respect to the given data points of the *S-N* curve.

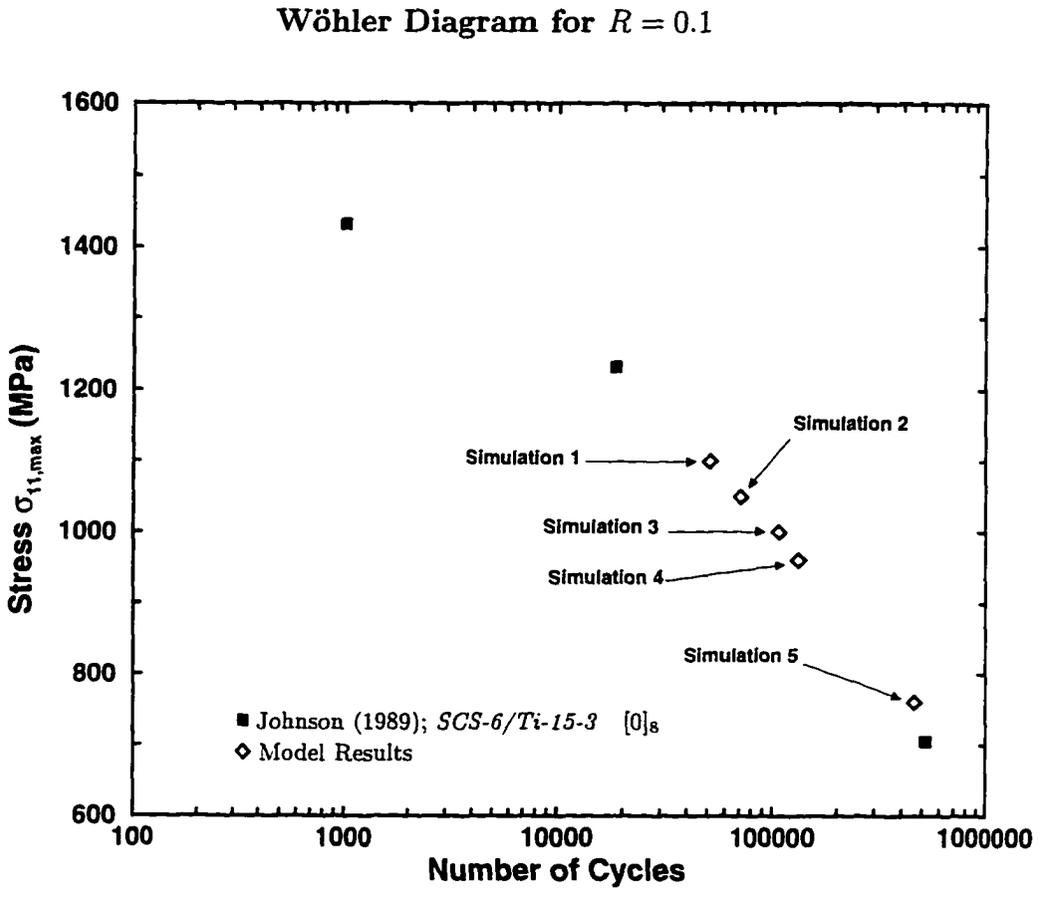

Figure 5.18: Comparison with experimental results (Johnson, 1989)

Chapter 6

Summary and Conclusions

A micro-mechanical damage model for high cycle fatigue loading based on thermo-dynamical principles is developed for uni-directional continuous fiber reinforced metal matrix composites. The model uses a micro-mechanical based theory to predict the fatigue damage evolution in the individual constituents of the composite material, namely the fibers and the matrix, from which the overall fatigue damage evolution in the composite is obtained. The micro-mechanical analysis is performed for each individual constituent using stress and strain concentration tensors based on the Mori-Tanaka method. A fatigue damage criterion based on thermo-dynamical principles is developed and applied to each of the constituents. Fatigue damage evolution equations are derived for the individual constituents and appropriate damage model parameters are established which reflect the physical behavior of the constituents with respect to damage evolution during the fatigue life of the material. The developed model is implemented into a numerical simulation code which is then used to simulate several fatigue tests for a uni-directional metal matrix composite system. High cycle fatigue loading is only modeled here, which is characterized by elastic deformations at the macroscale. The fatigue loading is applied as a uni-axial normal stress in the fiber direction in the form of a sinusoidal wave. A parametric study is conducted to show the influence of the model parameters on the damage evolution process in the constituents. Finally, several complete fatigue simulations are

performed. Results from these simulations are presented in the form of damage evolution curves for the individual constituents as well as the overall composite material. Furthermore, a comparison with available experimental data is made for the number of cycles to failure obtained from the simulations. This comparison is shown in the form of a *Wöhler diagram.*

6.1 Conclusions

The following conclusions are drawn from the presented work and the presented results.

1. A micro-mechanical fatigue damage theory for uni-directional metal matrix composites is developed based on thermo-dynamical principles. This model can simulate properly the behavior of matrix dominated as well as fiber dominated type composites.

2. A damage criterion is derived for each individual constituent together with the appropriate damage evolution equations. The criterion is based on parameters that are physically based.

3. Material and model parameters are established in a consistent manner where physical characteristics of the damage evolution processes in the individual constituents are incorporated.

4. The overall damage evolution in the composite material is obtained from the damage evolution in the individual constituents and can adequately represent the different types of damage modes due to fatigue in the composite.

5. Parametric studies on the damage evolution in the constituents with varying damage model parameters show the ability to model damage evolution in different uni-directional composite material systems based on the damage modes exhibited in experiments.

6. The results from the performed fatigue simulations in the form of damage evolution curves for the individual constituents show good qualitative behavior with respect to the physical observations made in fatigue experiments.

7. Results for the evolution of the effective volume fractions confirm the capability of the model to capture the physical behavior during the damage evolution process.

8. Comparison of the results for the number of cycles to final failure with those obtained from experiments is provided in the form of a *Wöhler diagram* and satisfactory agreement is shown.

6.2 Recommendations for Future Research

Recommendations for future research in this area are given as follows:

- Expand the model to include low-cycle fatigue for uni-directional metal matrix composites. Include plasticity effects through an appropriate cyclic plasticity model which addresses coupling effects between plasticity and damage.

- Expand the model to include fatigue behavior at elevated temperatures, where creep damage and environmental damage in form of oxidation contribute to the overall damage in the material. Include a visco-plasticity model to account for the visco-plastic as well as visco-elastic fatigue behavior at elevated temperatures.

- Establish an experimental program to generate the damage evolution curves from which the necessary model parameters are obtained to establish a wide fatigue loading spectrum.
- Expand the model to allow for multi-stress level fatigue loading in order to be able to model block loading or to approximate random fatigue loading.
- Adapt the model for applications to other fiber orientations.
- Adapt the model for applications to composite laminates of different ply stacking sequences.
- Incorporate the fatigue damage model into a finite element code to analyze real life composite structures.

References

S. M. Arnold and S. Kruch. A Differential CDM Model for Fatigue of Unidirectional Metal Matrix Composites. Technical Memorandum 105726, NASA, 1991a.

S. M. Arnold and S. Kruch. Differential Continuum Damage Mechanics Models for Creep and Fatigue of Unidirectional Metal Matrix Composites. Technical Memorandum 105213, NASA, 1991b.

S. M. Arnold and T. E. Wilt. A Deformation and Life Prediction of a Circumferentially Reinforced SiC/Ti-15-3 Ring. In Richard J. Schaller, editor, *Reliability, Stress Analysis and Failure Prevention*, pages 231–238. ASME, 1993. DE-Vol. 55.

Yakov Benveniste. A New Approach to the Application of Mori-Tanaka's Theory in Composite Materials. *Mechanics of Materials*, 6:147–157, 1987.

B. Budiansky. On the Elastic Moduli of some Heterogeneous Materials. *Journal of the Mechanics and Physics of Solids*, 13:223–227, 1965.

T. Bui-Quoc, J. Dubuc, A. Bazergui, and A. Biron. Cumulative Fatigue Damage under Stress Controlled Conditions. *Journal of Basic Engineering, Trans. ASME*, pages 691–698, 1971.

G. Cailletaud and C. Levaillant. Creep-Fatigue Life Prediction: What About Initiation? *Nucl. Engg. Design*, 83:279–292, 1984.

G. Cailletaud, H. Policella, and G. Baudin. Mesure de Déformation et d'Endommagement par Méthode Électrique. *La Recherche Aérospatiale*, 1980-1, 1980.

J.-L. Chaboche. Une loi différentielle d'endommagement de fatigue avec accumulation non-linéaire. *Revue Française de Mécanique*, (50-51), 1974. (English translation in *Mater. Building Res., Ann de l'ITBTP* 39 (1977)).

J.-L. Chaboche. *Description thermodynamique et phénoménologique de la viscoplasticité cyclique avec endommagement.* PhD thesis, Université P. et M. Curie, Paris, France, 1978.

J.-L. Chaboche. Le concept de contrainte effective applicqué à l'élasticité et à la viscoplasticité en présence d'un endommagement anisotrope. In Jean-Paul Boehler, editor, *Mechanical Behavior of Anisotropic Solids / N°295 Comportement*

Méchanique Des Solides Anisotropes. Martinus Nijhoff Publishers, 1979. Proceedings of the Euromech Colloquia 115, *N°*295 Villard-de-Lans, June 19-22, 1979, France.

J. L. Chaboche. Fracture Mechanics and Damage Mechanics: Complementarity of Approaches. In *Proceedings of the* 4^{th} *International Conference on* "NUMERICAL METHODS IN FRACTURE MECHANICS", pages 309–324, 1987.

J.-L. Chaboche. Continuum Damage Mechanics: Part I - General Concepts. *Journal Of Applied Mechanics*, 55:59–64, 1988a.

J. L. Chaboche. Continuum Damage Mechanics: Part II - Damage Growth, Crack Initiation and Crack Growth. *Journal of Applied Mechanics*, 55:65–72, 1988b.

J. L. Chaboche. Phenomenological Aspects of Continuum Damage Mechanics. In B. F. Dyson and D. P. Hayhurst, editors, MATERIALS AND ENGINEERING DESIGN: THE NEXT DECADE. Institute of Metals, 1989.

J. L. Chaboche and P. M. Lesne. A Non-Linear Continuous Fatigue Damage Model. *Fatigue and Fracture of Engineering Materials and Structures*, 11(1):1–17, 1988.

Jean-Louis Chaboche. Continuous Damage Mechanics - A Tool to Describe Phenomena Before Crack Initiation. *Nuclear Engineering and Design*, 64:233–247, 1981.

A. Charewicz and J. M. Daniel. Fatigue Damage Mechanisms and Residual Properties of Graphite/Epoxy Laminates. In *Symp. IUTAM on Mechanics of Damage and Fatigue, Haifa, Israel*, 1985.

T. Chen, George J. Dvorak, and Y. Benveniste. Mori-Tanaka Estimates of the Overall Elastic Moduli of Certain Composite Materials. *Journal of Applied Mechanics*, 59:539–546, 1992.

K. T. Chiang, D. H. Loh, P. K. Liaw, and E. S. Diaz. Materials Characterization of Silicon Carbide Reinforced Titanium $Ti/SCS-6$ Metal Matrix Composites: Part II. - Theoretical Modeling of Fatigue Behavior. *Metallurgical Transactions A*, 26A:3249–3255, 1995.

C. L. Chow and T. J. Lu. On Evolution Laws of Anisotropic Damage. *Engineering Fracture Mechanics*, 34(3):679–701, 1989.

C. L. Chow and Y. Wei. A Model of Continuum Damage Mechanics for Fatigue Failure. *International Journal of Fracture*, 50:301–316, 1991.

Demirkan Coker, Noel E. Ashbaugh, and Theodore Nicholas. Analysis of Thermomechanical Cyclic Behavior of Unidirectional Metal Matrix Composites. In H. Sehitoglu, editor, *Thermomechanical Fatigue Behavior of Materials,* ASTM STP 1186, pages 50–69. American Society for Testing and Materials, 1993.

J. P. Cordebois. *Critères d'Instabilité Plastique et Endommagement ductile en Grandes Deformations.* PhD thesis, Université Pierre et Marie Curie, 1983.

J. P. Cordebois and F. Sidoroff. Damage Induced Elastic Anisotropy. In Jean-Paul Boehler, editor, *Mechanical Behavior of Anisotropic Solids / N°295 Comportement Méchanique Des Solides Anisotropes.* Martinus Nijhoff Publishers, 1979. Proceedings of the Euromech Colloquia 115, *N°295* Villard-de-Lans, June 19-22, 1979, France.

H.T. Corten and T.I. Dolan. Cumulative Fatigue Damage. In *Proceedings of the International Conference on Fatigue of Metals.* IME and ASME, 1956.

George Dieter. *Mechanical Metallurgy.* McGraw-Hill, 1 edition, 1988.

George J. Dvorak and Y. A. Bahei-El-Din. Plasticity Analysis of Fibrous Composites. *Journal of Applied Mechanics*, 49:327–335, 1982.

V. D. Dyson and D. McLean. Creep of Nimonic 80A in Torsion and in Tension. *Met. Sci.*, 2:37pp., 1977.

A. C. Eringen. *Mechancis of Continua.* John Wiley and SOns, New York, 1967.

J. D. Eshelby. The Determination of the Elastic Fields of an Ellipsoidal Inclusion, and Related Problems. *Proceedings of the Royal Society London*, A 241:376–396, 1957.

J. T. Fong. What is Fatigue Damage? In K. L. Reifsnider, editor, *Damage in Composite Materials*, pages 243–266. American Society for Testing and Materials, Philadelphia, PA, 1982.

R.R. Gatts. Application of a Cumulative Damage Concept to Fatigue. *Transactions of the ASME*, pages 529–540, 1961.

J. E. Grady and B. A. Lerch. Evaluation of Thermomechanical Damage in Silicon Carbide/Titanium Composites. *AIAA Journal*, 29(6):992–997, 1991.

Gary R. Halford, Bradley A. Lerch, James F. Saltsman, and Vinod K. Arya. Proposed Framework for Thermomechanical Fatigue (TMF) Life Prediction of Metal Matrix Composites (MMC's). In H. Sehitoglu, editor, *Thermomechanical Fatigue Behavior of Materials,* ASTM STP 1186, pages 176–194. American Society for Testing and Materials, 1993.

K. A. Hart and S. Mall. Thermomechanical Fatigue Behavior of a Quasi-Isotropic $SCS-6/Ti-15-3$ Metal Matrix Composite. *Journal of Engineering Materials and Technology*, 117:109–117, 1995.

Z. Hashin. Cumulative Damage Theory for Composite Materials, Residual Life and Residual Strength Methods. *Composite Science and Technology*, 23:1–19, 1985.

D. L. Henry. A Theory of Fatigue-Damage Accumulation in Steel. *Transactions of the ASME*, pages 913–918, 1955.

R. Hill. A Self-Consistent Mechanics of Composite Materials. *Journal of the Mechanics and Physics of Solids*, 13:213–222, 1965.

C. T. Hua and D. F. Socie. Fatigue Dmage in 1045 Steel Under Constant Amplitude Biaxial Loading. *Fatigue of Engineering Materials and Structures*, 7(3):165–179, 1984.

W. Hwang and K. S. Han. Cumulative Fatigue Damage Models and Multi-Stress Fatigue Life Prediction. *Journal of Composite Materials*, 20:125–153, 1986a.

W. Hwang and K. S. Han. Fatigue of Composites - Fatigue Modulus Concept and Life Prediction. *Journal of Composite Materials*, 20:154–165, 1986b.

S. M. Jeng, P. Alassoeur, J. M. Yang, and S. Aksoy. Fracture Mechanics of Fiber-Reinforced Titanium Alloy Matrix Composites - Part IV: Low Cycle Fatigue. *Materials Science and Engineering*, A148:67–77, 1991a.

S. M. Jeng, J. M. Yang, and S. Aksoy. Damage Mechanisms of $SCS-6/Ti-6Al-4V$ Composites Under Thermo-Mechanical Fatigue. *Materials Science and Engineering*, A156:117–124, 1992.

S. M. Jeng, J. M. Yang, and C. J. Yang. Fracture Mechanics of Fiber-Reinforced Titanium Alloy Matrix Composites - Part II: Tensile Behavior. *Materials Science and Engineering*, A138:169–180, 1991b.

W. S. Johnson. Modeling Stiffness Loss in Boron / Aluminum Laminates below the Fatigue Limit. In T. K. O'Brien, editor, *Longterm Behavior of Composites*, ASTM STP 813, pages 160–176. ASTM, Philadelphia, PA, 1983.

W. S. Johnson. Fatigue Testing and Damage Development in Continuous Fiber Reinforced Metal Matrix Composites. In W. S. Johnson, editor, *Metal Matrix Composites: Testing, Analysis and Failure Modes*, pages 194–221. ASTM, Philadelphia, PA, 1989.

W. S. Johnson. Damage Development in Titanium Metal-Matrix Composites Subjected to Cyclic Loading. *Composites*, 24(3):187–196, 1993.

W. S. Johnson, M. Mirdamadi, and J. G. Bakuckas. Fatigue of Continuous Fiber Reinforced Metallic Materials. Technical Memorandum 107700, NASA, Langley Research Center, Hampton, Virginia 23681-0001, 1993.

W. S. Johnson and R. R. Wallis. Fatigue Behavior of Continuous Fiber Silicon-Carbide/Aluminum Composites. Technical Memorandum 86332, NASA, Langley Research Center, Hampton, VA, 1984a.

W. S. Johnson and R. R. Wallis. Fatigue Behavior of Continuous Fiber Silicon-Carbide/Aluminum Composites. In H. T. Hahn, editor, *Composite Materials: Fatigue and Fracture*, ASTM STP 907, pages 161–175. ASTM, Philadelphia, PA, 1984b.

J. J. Jonas and B. Baudelet. Effect of Crack and Cavity Generation on Tensile Stability. *Acta Metallurgica*, 25:43–50, 1977.

L. M. Kachanov. On the Creep Fracture Time. *Izv. Akad. Nauk. USSR Otd. Tekh.*, 8:26–31, 1958.

D. C. Lagoudas, A. C. Gavazzi, and H. Nigam. Elastoplastic Behavior of Metal Matrix Composites Based on Incremental Plasticity and the Mori-Tanaka Averaging Scheme. *Computational Mechanics*, 8:193–203, 1991.

J. Lemaitre and J. L. Chaboche. *Mechanics of Solids.* Cambridge University Press, 1990.

J. Lemaitre and A. Plumtree. Application of Damage Concepts to Predict Creep-Fatigue Failures. *Journal of Engineering Mechanics and Technology*, 101:284–292, 1979.

Jean Lemaitre. Evaluation of Dissipation and Damage in Metals. In *Proc. ICMI, Kyoto, Japan*, 1971.

Jean Lemaitre. How To Use Damage Mechanics. *Nuclear Engineering and Design*, 80:233–245, 1984.

Jean Lemaitre. A Continuous Damage Mechanics Model of Ductile Fracture. *Journal of Engineering Materials and Technology*, 107(42):83–89, 1985a.

Jean Lemaitre. Coupled Elasto-Plasticity and Damage Constitutive Equations. *Computer Methods in Applied Mechanics and Engineering*, 51:31–49, 1985b.

P. M. Lesne and S. Savalle. A Differential Damage Rule with Microinitiation and Micropropagation. *La Recherche Aérospatiale*, 1987(2):33–47, 1987.

P.M. Lesne and G. Cailletaud. Creep-Fatigue Interaction under High Frequency Loading. In *Int. Conf. on Mechanical Behavior of Materials, Beijing, China*, 1987.

C. Levaillant and A. Pineau. Assessment of High Temperature LCF and Life of Austenitic Stainless Steels by Using Intergranular Damage as a Correlating Parameter. In Amzallag, Leis, and Rabbe, editors, *LCF and Life Prediction*, ASTM-STP 770, pages 169–193. ASTM, 1982.

H. L. Leve. Cumulative Damage Theories. In A. F. Madayag, editor, *Metal Fatigue: Theory and Design*. J. Wiley, New York, 1960.

P. K. Liaw, E. S. Diaz, K. T. Chiang, and D. H. Loh. Materials Characterization of Silicon Carbide Reinforced Titanium $Ti/SCS-6$ Metal Matrix Composites: Part I. - Tensile and Fatigue Behavior. *Metallurgical Transactions A*, 26A:3225–3249, 1995.

Jacob Lubliner. *Plasticity Theory*. MacMillan Publishing Company, 1990.

S. S. Manson. Some Useful Concepts for the Designer in Treating Cumulative Damage at Elevated Temperature. volume 1, pages 13–45. I.C.M. 3, Cambridge, 1979.

S. M. Marco and W. L. Starkey. A Concept of Fatigue Damage. *Trans. ASME*, 76 (4):627–632, 1954.

J. Marin. *Mechanical Behavior of Engineering Materials*. Prentice-Hall, 1962.

Gerard A. Maugin. *The Thermomechanics of Plasticity and Fracture*. Cambridge University Press, 1992.

K. J. Miller and K. P. Zachariah. Cumulative Damage Laws for Fatigue Crack Initiation and Stage I Propagation. *Journal of Strain Analysis*, 12(4):262–270, 1977.

M. P. Miller, D. L. McDowell, R. L. T. Oehmke, and S. D. Antolovich. A Life Prediction Model for Thermomechanical Fatigue Based on Microcrack Propagation. In H. Sehitoglu, editor, *Thermomechanical Fatigue Behavior of Materials*, ASTM STP 1186, pages 35–49. American Society for Testing and Materials, 1993.

Milton A. Miner. Cumulative Damage in Fatigue. *Journal of Applied Mechanics*, 12:A 159–164, 1945.

M. Mirdamadi, W. S. Johnson, Y. A. Bahei-El-Din, and M. G. Castelli. Analysis of Thermomechanical Fatigue of Unidirectional Titanium Metal Matrix Composites. In W. W. Stinchcomb and N. E. Ashbaugh, editors, *Composite Materials: Fatigue*

and Fracture, Fourth Volume, pages 591–607. American Society for Testing and Materials, Philadelphia, PA, 1993.

T. Mori and K. Tanaka. Average Stress in Matrix and Average Elastic Energy of Materials with Misfitting Inclusions. *ACTA METALLURGICA*, 21:571–574, 1973.

T. Mura. *Micromechanics of Defects in Solids.* Martin Nijhoff, 2 edition, 1987.

S. Murakami. Effects of Cavity Distribution in Constitutive Equations of Creep and Creep Damage. In *Damage Mechanics*, 1981. Proceedings of the Euromech Colloquium 147, Cachan, France.

S. Murakami. Notion of Continuum Damage Mechanics and its Application to Anisotropic Creep Damage Thory. *Journal of Engineering Materials and Technology*, 105:99–, 1983.

S. Murakami. Mechanical Modeling of Material Damage. *Journal of Applied Mechanics*, 55, 1988.

S. Murakami and N. Ohno. A Continuum Theory of Creep and Creep Damage. In A. R. S. Ponter and D. R. Hayhurst, editors, *Creep in Structures*, 3rd *IUTAM Symposium on Creep in Structures*, Leicester, UK, 1980. Springer-Verlag.

R. W. Neu. A Mechanistic-Based Thermomechanical Fatigue Life Prediction Model For Metal Matrix Composites. *Fatigue and Fracture of Engineering Materials and Structures*, 16(8):811–828, 1993.

R. W. Neu and I. Roman. Acoustic Emission Monitoring of Damage in Metal Matrix Composites Subjected to Thermomechanical Fatigue. *Composite Science and Technology*, 52:1–8, 1994.

R. W. Neu and H. Sehitoglu. Thermomechanical Fatigue, Oxidation and Creep: Part I. Damage Mechanisms. *Metallurgical Transactions A*, 20 A:1755–1767, 1989a.

R. W. Neu and H. Sehitoglu. Thermomechanical Fatigue, Oxidation and Creep: Part II. Life Predicition. *Metallurgical Transactions A*, 20 A:1769–1783, 1989b.

T. Nicholas. Fatigue Life Prediction in Titanium Matrix Composites. *Journal of Engineering Materials and Technology*, 117:440–447, 1995.

T. Nicholas, J. L. Kroupa, and R. W. Neu. Analysis of a [0°/90°] Metal Matrix composite Under Thermo-Mechanical Fatigue Loading. *Composites Engineering*, 3(7-8):675–689, 1993.

A. N. Norris. An Examination of the Mori-Tanaka Effective Medium Approximation for Multiphase Composites. *Journal of Applied Mechanics*, 56:83–88, 1989.

M. J. Owen and R. J. Howe. The Accumulation of Damage in a Glass-reinforced Plastic under Tensile and Fatigue Loading. *Journal of Physics*, D:5:1637–1649, 1972.

A. Palmgren. *Ball and Roller Bearing Engineering.* (translated by G. Palmgren, et. al.), 1^{st} edition, 1945.

Arvid Palmgren. Die Lebensdauer von Kugellagern. *Zeitschrift des Vereines Deutscher Ingenieure*, 68:339–341, 1924.

Y. N. Rabotnov. Creep Rupture. In *Proceedings of the XII International Congree on Applied Mechanics*, pages 342–349. (Stanford-Springer, 1969), 1968.

S. M. Russ, T. Nicholas, M. Bates, and S. Mall. Thermomechanical Fatigue Behavior of SCS-6/Ti-24Al-11Nb Metal Matrix Composite. In *Failure Mechanisms in High Temperature Composite Materials*, AD-Vol. 22/AMD-Vol. 122, pages 91–106. ASME, 1991.

B. P. Sanders and S. Mall. Longitudinal Fatigue Response of a Metal Matrix Composite Under Strain Controlled Mode at Elevated Temperature. *Journal of Composites Technology & Research*, 16(4):304–313, 1994.

Walter Schütz. A History of Fatigue. *Engineering Fracture Mechanics*, 54(2):263–300, 1996.

F. Sidoroff. Description of Anisotropic Damage Application to Elasticity. In Jean-Paul Boehler, editor, *Mechanical Behavior of Anisotropic Solids / N°295 Comportement Méchanique Des Solides Anisotropes.* Martinus Nijhoff Publishers, 1979. Proceedings of the Euromech Colloquium 115, N°295 Villard-de-Lans, June 19-22, 1979, France.

F. Sidoroff. Description of Anisotropic Damage Application to Elasticity. In J. Hult and J. Lemaitre, editors, *Physical Non-Linearities in Structural Analysis*, IUTAM Series, pages 237–244. Springer-Verlag, 1980.

P. Srivatsavan and S. Subramanyan. A Cumulative Damage Rule Based on Successive Reduction in Fatigue Limit. *Journal of Engineering Mechanics and Technology*, 100:212–214, 1978.

M. Stumvoll and G. Swoboda. Deformation Behavior of Ductile Solids Containing Anisotropic Damage. *Journal of Engineering Mechanics*, 119(7):169–192, 1993.

S. Subramanyan. A Cumulative Damage Rule Based on the Knee Point of the *S-N*-Curve. *Journal of Engineering Mechanics and Technology*, pages 316–321, 1976.

Subra Suresh. *Fatigue of materials.* Cambridge Solid State Science Series. Cambridge University Press, 1991.

R. Talreja. *Fatigue of Composite Materials.* Technomic Publishing Co., Lancaster, PA, 1987.

Anthony R. Venson. *A Micro- and Macro- Experimental Investigation of Damage in Metal Matrix Composites.* PhD thesis, Louisiana State University, Baton Rouge, LA 70803, 1994.

G. Z. Voyiadjis and P. I. Kattan. A Coupled Thoery of Dmage Mechanics and Finite Strain Elasto-Plasticity, Part II: Damage and Finite Strain Plasticity. *International Journal of Engineering Science*, 28(6):505–524, 1990.

G. Z. Voyiadjis, P. I. Kattan, and A. R. Venson. Evolution of a Damage tensor for Metal Matrix Composites. In *MECAMAT 93, International Seminar on Micromechanics of Materials*, volume 84, pages 406–417, Moret-sur-Loing, France, 1993.

G. Z. Voyiadjis and T. Park. Anisotropic Damage Of Fiber Reinforced MMC Using An Overall Damage Analysis. *Journal of Engineering Mechanics*, 121(11):1209–1217, 1995a.

G. Z. Voyiadjis and T. Park. Local and Interfacial Damage Analysis of Metal Matrix Composites. *International Journal of Engineering Science*, 33(11):1595–1621, 1995b.

G. Z. Voyiadjis and A. R. Venson. Experimental Damage Investigation of a SiC-Ti-Aluminide Metal Matrix Composite. *International Journal of Damage Mechanics*, 4(4):338–361, 1995.

G. Z. Voyiadjis, A. R. Venson, and P. I. Kattan. Experimental Determination of Damage Parameters in Uni-axially Loaded Metal Matrix Composites Using the Overall Approach. *International Journal of Plasticity*, 11(8):895–926, 1996.

P. C. Wang, S. M. Jeng, J.-M. Yang, and S. M. Russ. Fatigue Damage Evolution and Property Degradation of a SCS-6/Ti-22Al-23Nb *Orthorombic* Titanium Aluminide Composite. *ACTA Materialia*, 44(8):3141–3156, 1996.

H. A. Whitworth. Cumulative Damage in Composites. *Transactions of the ASME*, 112:358–361, 1990.

J. R. Willis. Bounds and self-consistent estimates for the overall moduli of anisotrropic composites. *Journal of the Mechanics and Physics of Solids*, 25:185–202, 1977.

T. E. Wilt and S. M. Arnold. A Coupled/Uncoupled Deformation and Fatigue Damage Algorithm Utilizing the Finite Element Method. NASA TM 106526, NASA, Lewis Research Center, Cleveland, OH, 1994.

A. Wöhler. Resultate der in der Zentral-Werkstatt der Niederschlesisch-Märkischen Eisenbahn zu Frankfurt a.d.O. angestellten Versuche über die relative Festigkeit von Eisen, Stahl und Kupfer. *Zeitschrift für Bauwesen*, XVI:67–84, 1866.

A. Wöhler. Über die Festigkeitsversuche mit Eisen und Stahl. *Zeitschrift für Bauwesen*, XX:73–106, 1870.

D. A. Woodford. A Critical Assessment of the Fatigue Life Fraction Rule for Creep Rupture under Nonsteady Stress and Temperature. In *Int. Conf. on Creep Fatigue in Elevated Temperature Applications*, Philadelphia, PA, 1973.

Wen-Fang Wu, L. J. Lee, and S. T. Choi. A Study of Fatigue Damage and Fatigue Life of Composite Laminates. *Journal of Composite Materials*, 30(1):123–137, 1996.

Appendix A

Eshelby Tensor for Continuous Cylindrical Fibers

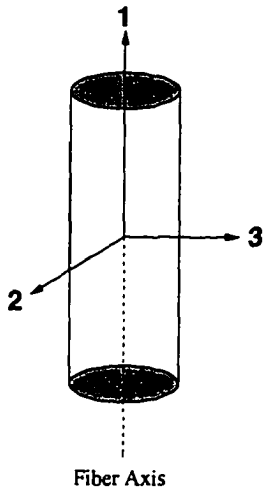

Figure A.1: Definition of the fiber axes for the Eshelby tensor of a continuous cylindrical fiber

An approximation for the Eshelby tensor based on the equivalent inclusion principle (Eshelby, 1957) is being given as (Mura, 1987)

$$
\begin{aligned}
S_{1111} &= 0 \\
S_{2222} &= \frac{1}{2(1-\nu)}\left[\frac{3}{4}+\frac{(1-2\nu)}{2}\right] \\
S_{3333} &= \frac{1}{2(1-\nu)}\left[\frac{3}{4}+\frac{(1-2\nu)}{2}\right] \\
S_{2233} &= \frac{1}{2(1-\nu)}\left[\frac{1}{4}-\frac{(1-2\nu)}{2}\right] \\
S_{3322} &= \frac{1}{2(1-\nu)}\left[\frac{1}{4}-\frac{(1-2\nu)}{2}\right]
\end{aligned}
$$

$$S_{3131} = \frac{1}{4}$$

$$S_{3311} = \frac{\nu}{2(1-\nu)}$$

$$S_{1122} = 0$$

$$S_{2211} = \frac{\nu}{2(1-\nu)}$$

$$S_{1133} = 0$$

$$S_{2323} = \frac{1}{2(1-\nu)}\left[\frac{1}{4} + \frac{(1-2\nu)}{2}\right]$$

$$S_{1212} = \frac{1}{4}$$

To obtain additional components for $\underline{\boldsymbol{S}}$ the symmetry condition

$$S_{ijkl} = S_{jikl} = S_{ijlk}$$

is employed. This yields

$$S_{1221} = S_{2112} = S_{2121} = S_{1212} = \frac{1}{4}$$

$$S_{1331} = S_{3113} = S_{1313} = S_{3131} = \frac{1}{4}$$

$$S_{2332} = S_{3223} = S_{3232} = S_{2323} = \frac{1}{2(1-\nu)}\left[\frac{1}{4} + \frac{(1-2\nu)}{2}\right]$$

All other tensor components of $\underline{\boldsymbol{S}}$ are equal to zero. The variable ν used in the above expressions represents Poisson's ratio for the matrix material which is assumed to be isotropic.

Appendix B

Derivation of Effective Volume Fractions

For the derivation of the general expression for the effective volume fractions (Voyiadjis and Park, 1995a) recourse is taken to a one-dimensional model used as a basis for the development of the effective stress concept in one dimenson (Rabotnov, 1968). The one-dimensional model is shown in principle in Figure B.1. The volume

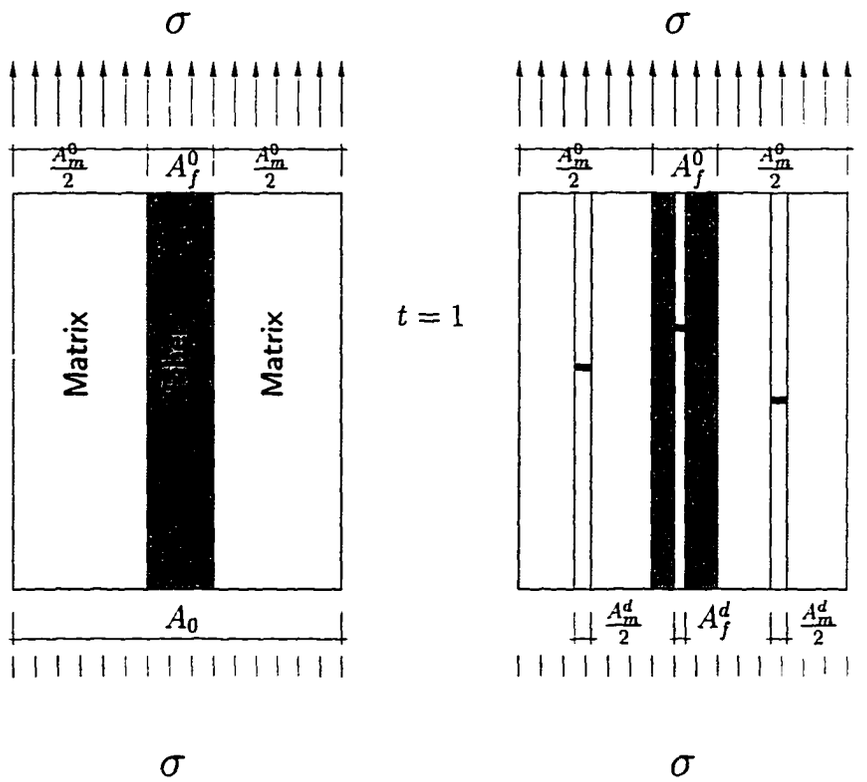

Figure B.1: Schematic of initial (undamaged) and damaged configurations for the derivation of the effective volume fractions

fractions in the initial configuration C_0 (Figure B.1) are defined as

$$c_f^0 = \frac{A_f^0}{A^0} \tag{B.1a}$$

$$c_m^0 = \frac{A_m^0}{A^0} \tag{B.1b}$$

satisfying

$$c_f^0 + c_m^0 = 1 \tag{B.2}$$

Considering the damaged configuration C one may establish similar relations for the volume fractions following the procedure used before for the undamaged configuration. Hence

$$\bar{c}_f = \frac{\bar{A}_f}{\bar{A}} \tag{B.3a}$$

$$\bar{c}_m = \frac{\bar{A}_m}{\bar{A}} \tag{B.3b}$$

satisfying

$$\bar{c}_f + \bar{c}_m = 1 \tag{B.4}$$

with $\bar{A}_f$ and $\bar{A}_m$ as the effective cross-sectional areas of the fiber and matrix, respectively, defined as

$$\bar{A}_f = A_f^0 - A_f^d \tag{B.5a}$$

$$\bar{A}_m = A_m^0 - A_m^d \tag{B.5b}$$

Here A_f^d and A_m^d represent the damaged portions of the fiber and matrix cross-sectional areas, respectively. But the damaged portion of the fiber and the matrix

maybe expressed using the damage variables ϕ_f and ϕ_m in one dimension as

$$A_f^d = \phi_f A_f^0 \tag{B.6a}$$

$$A_m^d = \phi_m A_m^0 \tag{B.6b}$$

Substitution of equations (B.6) into equations (B.5) yields

$$\bar{A}_f = A_f^0 - \phi_f A_f^0 = (1 - \phi_f) A_f^0 \tag{B.7a}$$

$$\bar{A}_m = A_m^0 - \phi_m A_m^0 = (1 - \phi_m) A_m^0 \tag{B.7b}$$

The quantity $\bar{A}$ in equations (B.3) represents the effective (damage free) cross-sectional area of the composite material and is defined as

$$\begin{aligned} \bar{A} = \bar{A}_f + \bar{A}_m &= \left(A_f^0 - A_f^d\right) + \left(A_m^0 - A_m^d\right) \\ &= (1 - \phi_f) A_f^0 + (1 - \phi_m) A_m^0 \end{aligned} \tag{B.8}$$

Backsubstitution of equations (B.7) and (B.8) into equations (B.3) allows one to derive an expression for the effective volume fractions at any loading state based on the initial volume fractions as

$$\bar{c}_f = \frac{(1 - \phi_f) A_f^0}{(1 - \phi_f) A_f^0 + (1 - \phi_m) A_m^0} \tag{B.9}$$

which is identical to

$$\bar{c}_f = \frac{(1 - \phi_f) \dfrac{A_f^0}{A^0}}{(1 - \phi_f) \dfrac{A_f^0}{A^0} + (1 - \phi_m) \dfrac{A_m^0}{A^0}} \tag{B.10}$$

Using equations (B.1) one obtains

$$\bar{c}_f = \frac{(1 - \phi_f) c_f^0}{(1 - \phi_f) c_f^0 + (1 - \phi_m) c_m^0} \tag{B.11}$$

or as a final expression for the effective fiber volume fraction $\bar{c}_f$

$$\bar{c}_f = \frac{(1-\phi_f)}{(1-\phi_f) + (1-\phi_m)\,\dfrac{c_m^0}{c_f^0}} \tag{B.12}$$

Similarly one obtains an expression for the effective matrix volume fraction $\bar{c}_m$ as

$$\bar{c}_m = \frac{(1-\phi_m)}{(1-\phi_m) + (1-\phi_f)\,\dfrac{c_f^0}{c_m^0}} \tag{B.13}$$

Expanding these expressions to three dimensions one needs to replace the damage variables ϕ_f and ϕ_m for the fiber and matrix by the equivalent damage variables ϕ_f^{eq} and ϕ_m^{eq} as defined in equations (3.57) to obtain the expressions for the effective volume fractions in three dimensions as given in equations (3.56).

Appendix C

Calculation of Partial Derivatives of the Damage Effect Tensor $\underline{\boldsymbol{M}}$

Using the relation between the applied Cauchy stress and the effective Cauchy stress as defined in equation (3.3) we can establish the damage effect tensor $\underline{\boldsymbol{M}}$ to obtain equation (3.4) with the components as defined in Section 3.2 by equations (3.5) to (3.22). Upon closer inspection of $\underline{\boldsymbol{M}}$ we see that the coefficients M_{ijkl} maybe written as the quotient of two tensor functions $\underline{\boldsymbol{F}}(\boldsymbol{\phi})$ and $g(\boldsymbol{\phi})$ such that

$$M_{ijkl} = \frac{F_{ijkl}}{2\,g} \tag{C.1}$$

where the components F_{ijkl} are given as

$$F_{1111} = 2\,(\chi_{22}\,\chi_{33} - \phi_{23}\phi_{32}) \qquad F_{1112} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31})$$

$$F_{1113} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}) \qquad F_{1121} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32})$$

$$F_{1131} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})$$

$$F_{1211} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \qquad F_{1212} = (\chi_{11}\,\chi_{33} + \chi_{22}\,\chi_{33} - \phi_{13}\,\phi_{31} - \phi_{23}\phi_{32})$$

$$F_{1213} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31}) \qquad F_{1222} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32})$$

$$F_{1232} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})$$

$$F_{1311} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23}) \qquad F_{1312} = (\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21})$$

$$F_{1323} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \qquad F_{1313} = (\chi_{11}\,\chi_{22} + \chi_{22}\,\chi_{33} - \phi_{12}\,\chi_{21} - \phi_{23}\phi_{32})$$

$$F_{1333} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})$$

$$F_{2111} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \qquad F_{2121} = (\chi_{11}\,\chi_{33} + \chi_{22}\,\chi_{33} - \phi_{23}\,\chi_{32} - \phi_{13}\phi_{31})$$

$$F_{2122} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \qquad F_{2123} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32})$$

$$F_{2131} = (\phi_{23}\,\chi_{11} + \phi_{21}\phi_{13})$$

$$F_{2212} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \qquad F_{2222} = 2\,(\chi_{11}\,\chi_{33} - \phi_{13}\phi_{31})$$

$$F_{2221} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \qquad F_{2223} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$

$$F_{2232} = (\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21}) \qquad \text{(C.2)}$$

$$F_{2313} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \qquad F_{2321} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23})$$

$$F_{2322} = (\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21}) \qquad F_{2323} = (\chi_{11}\,\chi_{22} + \chi_{11}\,\chi_{33} - \phi_{12}\,\phi_{21} - \phi_{13}\phi_{31})$$

$$F_{2333} = (\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21})$$

$$F_{3111} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}) \qquad F_{3121} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$

$$F_{3132} = (\phi_{21}\,\chi_{33} + \phi_{23}\phi_{31}) \qquad F_{3131} = (\chi_{11}\,\chi_{22} + \chi_{22}\,\chi_{33} - \phi_{12}\,\chi_{21} - \phi_{23}\phi_{32})$$

$$F_{3133} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32})$$

$$F_{3212} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}) \qquad F_{3222} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$

$$F_{3231} = (\phi_{12}\,\chi_{33} + \phi_{13}\phi_{32}) \qquad F_{3232} = (\chi_{11}\,\chi_{22} + \chi_{11}\,\chi_{33} - \phi_{12}\phi_{21} - \phi_{13}\phi_{31})$$

$$F_{3233} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$

$$F_{3313} = (\phi_{31}\,\chi_{22} + \phi_{21}\phi_{32}) \qquad F_{3323} = (\phi_{32}\,\chi_{11} + \phi_{12}\phi_{31})$$

$$F_{3331} = (\phi_{13}\,\chi_{22} + \phi_{12}\phi_{23}) \qquad F_{3332} = (\phi_{23}\,\chi_{11} + \phi_{13}\phi_{21})$$

$$F_{3333} = 2\,(\chi_{11}\,\chi_{22} - \phi_{12}\phi_{21})$$

and g the determinant of the tensor $(\delta_{ij} - \phi_{ij})$ as given in equation (3.23)

$$g = \chi_{11}\,\chi_{22}\,\chi_{33} - \chi_{11}\,\phi_{23}\,\phi_{32} - \chi_{22}\,\phi_{13}\,\phi_{31} - \chi_{33}\,\phi_{12}\,\phi_{21} - \phi_{12}\,\phi_{23}\,\phi_{31} - \phi_{13}\,\phi_{32}\,\phi_{21} \qquad \text{(C.3)}$$

All components F_{ijkl} not mentioned above are equal to zero. The variables χ_{11}, χ_{22} and χ_{33} are defined as

$$\begin{aligned} \chi_{11} &= \delta_{11} - \phi_{11} \\ \chi_{22} &= \delta_{22} - \phi_{22} \\ \chi_{33} &= \delta_{33} - \phi_{33} \end{aligned} \tag{C.4}$$

Using the definition for $\underline{\underline{M}}$ as given in equation (C.1) the partial derivatives of $\underline{\underline{M}}$ with respect to ϕ, as needed in the various equations in Section 3.4.4, are readily calculated using the quotient law of calculus. For the first partial derivative we obtain then

$$\frac{\partial M_{ijkl}}{\partial \phi_{mn}} = \frac{1}{2\,g}\frac{\partial F_{ijkl}}{\partial \phi_{mn}} - F_{ijkl}\frac{1}{2\,g^2}\frac{\partial g}{\partial \phi_{mn}} = \frac{1}{2\,g^2}\left(g\frac{\partial F_{ijkl}}{\partial \phi_{mn}} - F_{ijkl}\frac{\partial g}{\partial \phi_{mn}}\right) \tag{C.5}$$

and for the second partial derivative we obtain this way

$$\begin{aligned} \frac{\partial^2 M_{ijkl}}{\partial \phi_{mn}\,\partial \phi_{pq}} = \frac{1}{2\,g}\frac{\partial^2 F_{ijkl}}{\partial \phi_{mn}\,\partial \phi_{pq}} &- \frac{1}{2\,g^2}\frac{\partial g}{\partial \phi_{pq}}\frac{\partial F_{ijkl}}{\partial \phi_{mn}} - \frac{\partial F_{ijkl}}{\partial \phi_{pq}}\frac{1}{2\,g^2}\frac{\partial g}{\partial \phi_{mn}} \\ &- F_{ijkl}\frac{(-2)}{2\,g^3}\frac{\partial g}{\partial \phi_{pq}}\frac{\partial g}{\partial \phi_{mn}} - F_{ijkl}\frac{1}{2\,g^2}\frac{\partial^2 g}{\partial \phi_{mn}\,\partial \phi_{pq}} \end{aligned} \tag{C.6}$$

After rearrangement we obtain the final expression as

$$\begin{aligned} \frac{\partial^2 M_{ijkl}}{\partial \phi_{mn}\partial \phi_{pq}} = \frac{1}{2\,g^3}\frac{\partial g}{\partial \phi_{pq}}&\left(2\,F_{ijkl}\frac{\partial g}{\partial \phi_{mn}} - g\frac{\partial F_{ijkl}}{\partial \phi_{mn}}\right) \\ &+ \frac{1}{2\,g^2}\left(g\frac{\partial^2 F_{ijkl}}{\partial \phi_{mn}\partial \phi_{pq}} - \frac{\partial F_{ijkl}}{\partial \phi_{pq}}\frac{\partial g}{\partial \phi_{mn}} - F_{ijkl}\frac{\partial^2 g}{\partial \phi_{mn}\,\partial \phi_{pq}}\right) \end{aligned} \tag{C.7}$$

The individual components for the partial derivatives $\frac{\partial g}{\partial \phi_{mn}}$, $\frac{\partial^2 g}{\partial \phi_{mn}\,\partial \phi_{pq}}$, $\frac{\partial F_{ijkl}}{\partial \phi_{mn}}$ and $\frac{\partial^2 F_{ijkl}}{\partial \phi_{mn}\,\partial \phi_{pq}}$ are given as shown in the following.

The first partial derivatives of g with respect to ϕ_{mn}:

$$\begin{aligned}
\frac{\partial g}{\partial \phi_{11}} &= -\chi_{22} * \chi_{33} + \phi_{23} * \phi_{32} \\
\frac{\partial g}{\partial \phi_{12}} &= -\phi_{21} * \chi_{33} - \phi_{23} * \phi_{31} \\
\frac{\partial g}{\partial \phi_{13}} &= -\phi_{31} * \chi_{22} - \phi_{21} * \phi_{32} \\
\frac{\partial g}{\partial \phi_{21}} &= -\phi_{12} * \chi_{33} - \phi_{32} * \phi_{13} \\
\frac{\partial g}{\partial \phi_{22}} &= -\chi_{11} * \chi_{33} + \phi_{13} * \phi_{31} \\
\frac{\partial g}{\partial \phi_{23}} &= -\phi_{32} * \chi_{11} - \phi_{12} * \phi_{31} \\
\frac{\partial g}{\partial \phi_{31}} &= -\phi_{13} * \chi_{22} - \phi_{12} * \phi_{23} \\
\frac{\partial g}{\partial \phi_{32}} &= -\phi_{23} * \chi_{11} - \phi_{21} * \phi_{13} \\
\frac{\partial g}{\partial \phi_{33}} &= -\chi_{11} * \chi_{22} + \phi_{12} * \phi_{21}
\end{aligned} \tag{C.8}$$

The second partial derivatives of g with respect to $\phi_{mn}\,\phi_{pq}$:

$$\begin{aligned}
&\frac{\partial^2 g}{\partial \phi_{11}\,\partial \phi_{22}} = \chi_{33} \qquad \frac{\partial^2 g}{\partial \phi_{11}\,\partial \phi_{33}} = \chi_{22} \qquad \frac{\partial^2 g}{\partial \phi_{11}\,\partial \phi_{23}} = \phi_{32} \\
&\frac{\partial^2 g}{\partial \phi_{11}\,\partial \phi_{32}} = \phi_{23} \qquad \frac{\partial^2 g}{\partial \phi_{12}\,\partial \phi_{21}} = -\chi_{33} \qquad \frac{\partial^2 g}{\partial \phi_{12}\,\partial \phi_{33}} = \phi_{12} \\
&\frac{\partial^2 g}{\partial \phi_{12}\,\partial \phi_{23}} = -\phi_{31} \qquad \frac{\partial^2 g}{\partial \phi_{12}\,\partial \phi_{31}} = -\phi_{23} \qquad \frac{\partial^2 g}{\partial \phi_{13}\,\partial \phi_{22}} = \phi_{31} \\
&\frac{\partial^2 g}{\partial \phi_{13}\,\partial \phi_{21}} = -\phi_{32} \qquad \frac{\partial^2 g}{\partial \phi_{13}\,\partial \phi_{32}} = -\phi_{21} \qquad \frac{\partial^2 g}{\partial \phi_{13}\,\partial \phi_{31}} = -\chi_{22} \\
&\frac{\partial^2 g}{\partial \phi_{21}\,\partial \phi_{33}} = \phi_{12} \qquad \frac{\partial^2 g}{\partial \phi_{21}\,\partial \phi_{12}} = -\chi_{33} \qquad \frac{\partial^2 g}{\partial \phi_{21}\,\partial \phi_{13}} = -\phi_{32} \\
&\frac{\partial^2 g}{\partial \phi_{21}\,\partial \phi_{32}} = -\phi_{13} \qquad \frac{\partial^2 g}{\partial \phi_{22}\,\partial \phi_{11}} = \chi_{33} \qquad \frac{\partial^2 g}{\partial \phi_{22}\,\partial \phi_{13}} = \phi_{31} \\
&\frac{\partial^2 g}{\partial \phi_{22}\,\partial \phi_{31}} = \phi_{13} \qquad \frac{\partial^2 g}{\partial \phi_{22}\,\partial \phi_{33}} = \chi_{11} \qquad \frac{\partial^2 g}{\partial \phi_{23}\,\partial \phi_{11}} = \phi_{32} \\
&\frac{\partial^2 g}{\partial \phi_{23}\,\partial \phi_{12}} = -\phi_{31} \qquad \frac{\partial^2 g}{\partial \phi_{23}\,\partial \phi_{32}} = -\chi_{11} \qquad \frac{\partial^2 g}{\partial \phi_{23}\,\partial \phi_{31}} = -\phi_{12}
\end{aligned} \tag{C.9}$$

$$\frac{\partial^2 g}{\partial\phi_{31}\,\partial\phi_{22}} = \phi_{13} \qquad \frac{\partial^2 g}{\partial\phi_{31}\,\partial\phi_{13}} = -\chi_{22} \qquad \frac{\partial^2 g}{\partial\phi_{31}\,\partial\phi_{12}} = -\phi_{23}$$
$$\frac{\partial^2 g}{\partial\phi_{31}\,\partial\phi_{23}} = -\phi_{12} \qquad \frac{\partial^2 g}{\partial\phi_{32}\,\partial\phi_{11}} = \phi_{23} \qquad \frac{\partial^2 g}{\partial\phi_{32}\,\partial\phi_{23}} = -\chi_{11}$$
$$\frac{\partial^2 g}{\partial\phi_{32}\,\partial\phi_{21}} = -\phi_{13} \qquad \frac{\partial^2 g}{\partial\phi_{32}\,\partial\phi_{13}} = -\phi_{21} \qquad \frac{\partial^2 g}{\partial\phi_{33}\,\partial\phi_{11}} = \chi_{22}$$
$$\frac{\partial^2 g}{\partial\phi_{33}\,\partial\phi_{22}} = \chi_{11} \qquad \frac{\partial^2 g}{\partial\phi_{33}\,\partial\phi_{12}} = \phi_{21} \qquad \frac{\partial^2 g}{\partial\phi_{33}\,\partial\phi_{21}} = \phi_{12}$$

The first partial derivatives of F_{ijkl} with respect to ϕ_{mn}

$$\frac{\partial F_{1111}}{\partial\phi_{22}} = -2\,\chi_{33} \qquad \frac{\partial F_{1111}}{\partial\phi_{33}} = -2\,\chi_{22} \qquad \frac{\partial F_{1111}}{\partial\phi_{23}} = -2\,\phi_{32}$$
$$\frac{\partial F_{1111}}{\partial\phi_{32}} = -2\,\phi_{23} \qquad \frac{\partial F_{1112}}{\partial\phi_{33}} = -\phi_{21} \qquad \frac{\partial F_{1112}}{\partial\phi_{21}} = \chi_{33}$$
$$\frac{\partial F_{1112}}{\partial\phi_{23}} = \phi_{31} \qquad \frac{\partial F_{1112}}{\partial\phi_{31}} = \phi_{23} \qquad \frac{\partial F_{1113}}{\partial\phi_{22}} = -\phi_{31}$$
$$\frac{\partial F_{1113}}{\partial\phi_{21}} = \phi_{32} \qquad \frac{\partial F_{1113}}{\partial\phi_{32}} = \phi_{21} \qquad \frac{\partial F_{1113}}{\partial\phi_{31}} = \chi_{22}$$
$$\frac{\partial F_{1121}}{\partial\phi_{33}} = -\phi_{12} \qquad \frac{\partial F_{1121}}{\partial\phi_{12}} = \chi_{33} \qquad \frac{\partial F_{1121}}{\partial\phi_{32}} = \phi_{13}$$
$$\frac{\partial F_{1121}}{\partial\phi_{13}} = \phi_{32} \qquad \frac{\partial F_{1131}}{\partial\phi_{22}} = -\phi_{13} \qquad \frac{\partial F_{1131}}{\partial\phi_{12}} = \phi_{23}$$
$$\frac{\partial F_{1131}}{\partial\phi_{23}} = \phi_{12} \qquad \frac{\partial F_{1131}}{\partial\phi_{13}} = \chi_{22}$$
$$\frac{\partial F_{1211}}{\partial\phi_{33}} = -\phi_{12} \qquad \frac{\partial F_{1211}}{\partial\phi_{12}} = \chi_{33} \qquad \frac{\partial F_{1211}}{\partial\phi_{32}} = \phi_{13}$$
$$\frac{\partial F_{1211}}{\partial\phi_{13}} = \phi_{32} \qquad \frac{\partial F_{1212}}{\partial\phi_{11}} = -\chi_{33} \qquad \frac{\partial F_{1212}}{\partial\phi_{22}} = -\chi_{33}$$
$$\frac{\partial F_{1212}}{\partial\phi_{33}} = -\chi_{22} - \chi_{11} \qquad \frac{\partial F_{1212}}{\partial\phi_{31}} = -\phi_{13} \qquad \frac{\partial F_{1212}}{\partial\phi_{13}} = -\phi_{31}$$
$$\frac{\partial F_{1212}}{\partial\phi_{32}} = -\phi_{23} \qquad \frac{\partial F_{1212}}{\partial\phi_{23}} = -\phi_{32} \qquad \frac{\partial F_{1213}}{\partial\phi_{11}} = -\phi_{32}$$
$$\frac{\partial F_{1213}}{\partial\phi_{12}} = \phi_{31} \qquad \frac{\partial F_{1213}}{\partial\phi_{32}} = \chi_{11} \qquad \frac{\partial F_{1213}}{\partial\phi_{31}} = \phi_{12}$$
$$\frac{\partial F_{1222}}{\partial\phi_{33}} = -\phi_{12} \qquad \frac{\partial F_{1222}}{\partial\phi_{12}} = \chi_{33} \qquad \frac{\partial F_{1222}}{\partial\phi_{32}} = \phi_{13}$$
$$\frac{\partial F_{1222}}{\partial\phi_{13}} = \phi_{32} \qquad \frac{\partial F_{1232}}{\partial\phi_{22}} = -\phi_{13} \qquad \frac{\partial F_{1232}}{\partial\phi_{12}} = \phi_{23}$$

$$
\begin{array}{lll}
\frac{\partial F_{1232}}{\partial \phi_{23}} = \phi_{12} & \frac{\partial F_{1232}}{\partial \phi_{13}} = \chi_{22} & \\
\frac{\partial F_{1311}}{\partial \phi_{22}} = -\phi_{13} & \frac{\partial F_{1311}}{\partial \phi_{12}} = \phi_{23} & \frac{\partial F_{1311}}{\partial \phi_{23}} = \phi_{12} \\
\frac{\partial F_{1311}}{\partial \phi_{13}} = \chi_{22} & \frac{\partial F_{1312}}{\partial \phi_{11}} = -\phi_{23} & \frac{\partial F_{1312}}{\partial \phi_{21}} = \phi_{13} \\
\frac{\partial F_{1312}}{\partial \phi_{23}} = \chi_{11} & \frac{\partial F_{1312}}{\partial \phi_{13}} = \phi_{21} & \frac{\partial F_{1313}}{\partial \phi_{11}} = -\chi_{22} \\
\frac{\partial F_{1313}}{\partial \phi_{22}} = -\chi_{11} - \chi_{33} & \frac{\partial F_{1313}}{\partial \phi_{33}} = -\chi_{22} & \frac{\partial F_{1313}}{\partial \phi_{12}} = -\phi_{21} \\
\frac{\partial F_{1313}}{\partial \phi_{21}} = -\phi_{12} & \frac{\partial F_{1313}}{\partial \phi_{23}} = -\phi_{32} & \frac{\partial F_{1313}}{\partial \phi_{32}} = -\phi_{23} \\
\frac{\partial F_{1323}}{\partial \phi_{33}} = -\phi_{12} & \frac{\partial F_{1323}}{\partial \phi_{12}} = \chi_{33} & \frac{\partial F_{1323}}{\partial \phi_{32}} = \phi_{13} \\
\frac{\partial F_{1323}}{\partial \phi_{13}} = \phi_{32} & \frac{\partial F_{1333}}{\partial \phi_{22}} = -\phi_{13} & \frac{\partial F_{1333}}{\partial \phi_{12}} = \phi_{23} \\
\frac{\partial F_{1333}}{\partial \phi_{23}} = \phi_{12} & \frac{\partial F_{1333}}{\partial \phi_{13}} = \chi_{22} & \\
\frac{\partial F_{2111}}{\partial \phi_{33}} = -\phi_{21} & \frac{\partial F_{2111}}{\partial \phi_{21}} = \chi_{33} & \frac{\partial F_{2111}}{\partial \phi_{23}} = \phi_{31} \\
\frac{\partial F_{2111}}{\partial \phi_{31}} = \phi_{23} & \frac{\partial F_{2121}}{\partial \phi_{11}} = -\chi_{33} & \frac{\partial F_{2121}}{\partial \phi_{22}} = -\chi_{33} \\
\frac{\partial F_{2121}}{\partial \phi_{33}} = -\chi_{11} - \chi_{22} & \frac{\partial F_{2121}}{\partial \phi_{13}} = -\phi_{31} & \frac{\partial F_{2121}}{\partial \phi_{31}} = -\phi_{13} \\
\frac{\partial F_{2121}}{\partial \phi_{23}} = -\phi_{32} & \frac{\partial F_{2121}}{\partial \phi_{32}} = -\phi_{23} & \frac{\partial F_{2122}}{\partial \phi_{33}} = -\phi_{21} \\
\frac{\partial F_{2122}}{\partial \phi_{21}} = \chi_{33} & \frac{\partial F_{2122}}{\partial \phi_{23}} = \phi_{31} & \frac{\partial F_{2122}}{\partial \phi_{31}} = \phi_{23} \\
\frac{\partial F_{2123}}{\partial \phi_{22}} = -\phi_{31} & \frac{\partial F_{2123}}{\partial \phi_{31}} = \chi_{22} & \frac{\partial F_{2123}}{\partial \phi_{32}} = \phi_{21} \\
\frac{\partial F_{2123}}{\partial \phi_{21}} = \phi_{32} & \frac{\partial F_{2131}}{\partial \phi_{11}} = -\phi_{23} & \frac{\partial F_{2131}}{\partial \phi_{23}} = \chi_{11} \\
\frac{\partial F_{2131}}{\partial \phi_{21}} = \phi_{13} & \frac{\partial F_{2131}}{\partial \phi_{13}} = \phi_{21} & \\
\frac{\partial F_{2212}}{\partial \phi_{33}} = -\phi_{21} & \frac{\partial F_{2212}}{\partial \phi_{21}} = \chi_{33} & \frac{\partial F_{2212}}{\partial \phi_{23}} = \phi_{31} \\
\frac{\partial F_{2212}}{\partial \phi_{31}} = \phi_{23} & \frac{\partial F_{2222}}{\partial \phi_{11}} = -2\,\chi_{33} & \frac{\partial F_{2222}}{\partial \phi_{33}} = -2\,\chi_{11}
\end{array}
\tag{C.10}
$$

$$\frac{\partial F_{2222}}{\partial \phi_{31}} = -2\,\phi_{13} \qquad \frac{\partial F_{2222}}{\partial \phi_{13}} = -2\,\phi_{31} \qquad \frac{\partial F_{2221}}{\partial \phi_{33}} = -\phi_{12}$$

$$\frac{\partial F_{2221}}{\partial \phi_{12}} = \chi_{33} \qquad \frac{\partial F_{2221}}{\partial \phi_{13}} = \phi_{32} \qquad \frac{\partial F_{2221}}{\partial \phi_{32}} = \phi_{13}$$

$$\frac{\partial F_{2223}}{\partial \phi_{11}} = -\phi_{32} \qquad \frac{\partial F_{2223}}{\partial \phi_{32}} = \chi_{11} \qquad \frac{\partial F_{2223}}{\partial \phi_{12}} = \phi_{31}$$

$$\frac{\partial F_{2223}}{\partial \phi_{31}} = \phi_{12} \qquad \frac{\partial F_{2232}}{\partial \phi_{11}} = -\phi_{23} \qquad \frac{\partial F_{2232}}{\partial \phi_{23}} = \chi_{11}$$

$$\frac{\partial F_{2232}}{\partial \phi_{21}} = \phi_{13} \qquad \frac{\partial F_{2232}}{\partial \phi_{13}} = \phi_{21}$$

$$\frac{\partial F_{2313}}{\partial \phi_{33}} = -\phi_{21} \qquad \frac{\partial F_{2313}}{\partial \phi_{21}} = \chi_{33} \qquad \frac{\partial F_{2313}}{\partial \phi_{23}} = \phi_{31}$$

$$\frac{\partial F_{2313}}{\partial \phi_{31}} = \phi_{23} \qquad \frac{\partial F_{2321}}{\partial \phi_{22}} = -\phi_{13} \qquad \frac{\partial F_{2321}}{\partial \phi_{13}} = \chi_{22}$$

$$\frac{\partial F_{2321}}{\partial \phi_{12}} = \phi_{23} \qquad \frac{\partial F_{2321}}{\partial \phi_{23}} = \phi_{12} \qquad \frac{\partial F_{2322}}{\partial \phi_{11}} = -\phi_{23}$$

$$\frac{\partial F_{2322}}{\partial \phi_{23}} = \chi_{11} \qquad \frac{\partial F_{2322}}{\partial \phi_{13}} = \phi_{21} \qquad \frac{\partial F_{2322}}{\partial \phi_{21}} = \phi_{13}$$

$$\frac{\partial F_{2323}}{\partial \phi_{11}} = -\chi_{33} - \chi_{22} \qquad \frac{\partial F_{2323}}{\partial \phi_{22}} = -\chi_{11} \qquad \frac{\partial F_{2323}}{\partial \phi_{33}} = -\chi_{11}$$

$$\frac{\partial F_{2323}}{\partial \phi_{12}} = -\phi_{21} \qquad \frac{\partial F_{2323}}{\partial \phi_{21}} = -\phi_{12} \qquad \frac{\partial F_{2323}}{\partial \phi_{31}} = -\phi_{13}$$

$$\frac{\partial F_{2323}}{\partial \phi_{13}} = -\phi_{31} \qquad \frac{\partial F_{2333}}{\partial \phi_{11}} = -\phi_{23} \qquad \frac{\partial F_{2333}}{\partial \phi_{23}} = \chi_{11}$$

$$\frac{\partial F_{2333}}{\partial \phi_{13}} = \phi_{21} \qquad \frac{\partial F_{2333}}{\partial \phi_{21}} = \phi_{13}$$

$$\frac{\partial F_{3111}}{\partial \phi_{22}} = -\phi_{31} \qquad \frac{\partial F_{3111}}{\partial \phi_{31}} = \chi_{22} \qquad \frac{\partial F_{3111}}{\partial \phi_{21}} = \phi_{32}$$

$$\frac{\partial F_{3111}}{\partial \phi_{32}} = \phi_{21} \qquad \frac{\partial F_{3121}}{\partial \phi_{11}} = -\phi_{32} \qquad \frac{\partial F_{3121}}{\partial \phi_{32}} = \chi_{11}$$

$$\frac{\partial F_{3121}}{\partial \phi_{12}} = \phi_{31} \qquad \frac{\partial F_{3121}}{\partial \phi_{31}} = \phi_{12} \qquad \frac{\partial F_{3131}}{\partial \phi_{11}} = -\chi_{22}$$

$$\frac{\partial F_{3131}}{\partial \phi_{22}} = -\chi_{33} - \chi_{11} \qquad \frac{\partial F_{3131}}{\partial \phi_{33}} = -\chi_{22} \qquad \frac{\partial F_{3131}}{\partial \phi_{12}} = -\phi_{21}$$

$$\frac{\partial F_{3131}}{\partial \phi_{21}} = -\phi_{12} \qquad \frac{\partial F_{3131}}{\partial \phi_{23}} = -\phi_{32} \qquad \frac{\partial F_{3131}}{\partial \phi_{32}} = -\phi_{23}$$

$$\frac{\partial F_{3132}}{\partial \phi_{33}} = -\phi_{21} \qquad \frac{\partial F_{3132}}{\partial \phi_{21}} = \chi_{33} \qquad \frac{\partial F_{3132}}{\partial \phi_{23}} = \phi_{31}$$

$$\frac{\partial F_{3132}}{\partial \phi_{31}} = \phi_{23} \qquad \frac{\partial F_{3133}}{\partial \phi_{22}} = -\phi_{31} \qquad \frac{\partial F_{3133}}{\partial \phi_{31}} = \chi_{22}$$

$$\frac{\partial F_{3133}}{\partial \phi_{21}} = \phi_{32} \qquad \frac{\partial F_{3133}}{\partial \phi_{32}} = \phi_{21}$$

$$\frac{\partial F_{3212}}{\partial \phi_{22}} = -\phi_{31} \qquad \frac{\partial F_{3212}}{\partial \phi_{31}} = \chi_{22} \qquad \frac{\partial F_{3212}}{\partial \phi_{21}} = \phi_{32}$$

$$\frac{\partial F_{3212}}{\partial \phi_{32}} = \phi_{21} \qquad \frac{\partial F_{3222}}{\partial \phi_{11}} = -\phi_{32} \qquad \frac{\partial F_{3222}}{\partial \phi_{32}} = \chi_{11}$$

$$\frac{\partial F_{3222}}{\partial \phi_{12}} = \phi_{31} \qquad \frac{\partial F_{3222}}{\partial \phi_{31}} = \phi_{12} \qquad \frac{\partial F_{3231}}{\partial \phi_{33}} = -\phi_{12}$$

$$\frac{\partial F_{3231}}{\partial \phi_{12}} = \chi_{33} \qquad \frac{\partial F_{3231}}{\partial \phi_{13}} = \phi_{32} \qquad \frac{\partial F_{3231}}{\partial \phi_{32}} = \phi_{13}$$

$$\frac{\partial F_{3232}}{\partial \phi_{11}} = -\chi_{22} - \chi_{33} \qquad \frac{\partial F_{3232}}{\partial \phi_{22}} = -\chi_{11} \qquad \frac{\partial F_{3232}}{\partial \phi_{33}} = -\chi_{11}$$

$$\frac{\partial F_{3232}}{\partial \phi_{12}} = -\phi_{21} \qquad \frac{\partial F_{3232}}{\partial \phi_{21}} = -\phi_{12} \qquad \frac{\partial F_{3232}}{\partial \phi_{13}} = -\phi_{31}$$

$$\frac{\partial F_{3232}}{\partial \phi_{31}} = -\phi_{13} \qquad \frac{\partial F_{3233}}{\partial \phi_{11}} = -\phi_{32} \qquad \frac{\partial F_{3233}}{\partial \phi_{32}} = \chi_{11}$$

$$\frac{\partial F_{3233}}{\partial \phi_{12}} = \phi_{31} \qquad \frac{\partial F_{3233}}{\partial \phi_{31}} = \phi_{12}$$

$$\frac{\partial F_{3313}}{\partial \phi_{22}} = -\phi_{31} \qquad \frac{\partial F_{3313}}{\partial \phi_{31}} = \chi_{22} \qquad \frac{\partial F_{3313}}{\partial \phi_{21}} = \phi_{32}$$

$$\frac{\partial F_{3313}}{\partial \phi_{32}} = \phi_{21} \qquad \frac{\partial F_{3323}}{\partial \phi_{11}} = -\phi_{32} \qquad \frac{\partial F_{3323}}{\partial \phi_{32}} = \chi_{11}$$

$$\frac{\partial F_{3323}}{\partial \phi_{12}} = \phi_{31} \qquad \frac{\partial F_{3323}}{\partial \phi_{31}} = \phi_{12} \qquad \frac{\partial F_{3333}}{\partial \phi_{11}} = -2\,\chi_{22}$$

$$\frac{\partial F_{3333}}{\partial \phi_{22}} = -2\,\chi_{11} \qquad \frac{\partial F_{3333}}{\partial \phi_{12}} = -2\,\phi_{21} \qquad \frac{\partial F_{3333}}{\partial \phi_{21}} = -2\,\phi_{12}$$

$$\frac{\partial F_{3331}}{\partial \phi_{22}} = -\phi_{13} \qquad \frac{\partial F_{3331}}{\partial \phi_{13}} = \chi_{22} \qquad \frac{\partial F_{3331}}{\partial \phi_{12}} = \phi_{23}$$

$$\frac{\partial F_{3331}}{\partial \phi_{23}} = \phi_{12} \qquad \frac{\partial F_{3332}}{\partial \phi_{11}} = -\phi_{23} \qquad \frac{\partial F_{3332}}{\partial \phi_{23}} = \chi_{11}$$

$$\frac{\partial F_{3332}}{\partial \phi_{13}} = \phi_{21} \qquad \frac{\partial F_{3332}}{\partial \phi_{21}} = \phi_{13}$$

All other partial derivatives not explicit defined are equal to zero.

The second partial derivatives of F_{ijkl} with respect to $\phi_{mn}\,\phi_{pq}$:

$$\frac{\partial^2 F_{1111}}{\partial\phi_{22}\,\partial\phi_{33}} = 2 \qquad \frac{\partial^2 F_{1111}}{\partial\phi_{33}\,\partial\phi_{22}} = 2 \qquad \frac{\partial^2 F_{1111}}{\partial\phi_{23}\,\partial\phi_{32}} = -2$$

$$\frac{\partial^2 F_{1111}}{\partial\phi_{32}\,\partial\phi_{23}} = -2 \qquad \frac{\partial^2 F_{1112}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{1112}}{\partial\phi_{21}\,\partial\phi_{33}} = -1$$

$$\frac{\partial^2 F_{1112}}{\partial\phi_{23}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{1112}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{1113}}{\partial\phi_{22}\,\partial\phi_{31}} = -1$$

$$\frac{\partial^2 F_{1113}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{1113}}{\partial\phi_{21}\,\partial\phi_{32}} = 1 \qquad \frac{\partial^2 F_{1113}}{\partial\phi_{32}\,\partial\phi_{21}} = 1$$

$$\frac{\partial^2 F_{1121}}{\partial\phi_{33}\,\partial\phi_{12}} = -1 \qquad \frac{\partial^2 F_{1121}}{\partial\phi_{12}\,\partial\phi_{33}} = -1 \qquad \frac{\partial^2 F_{1121}}{\partial\phi_{13}\,\partial\phi_{32}} = 1$$

$$\frac{\partial^2 F_{1121}}{\partial\phi_{32}\,\partial\phi_{13}} = 1 \qquad \frac{\partial^2 F_{1131}}{\partial\phi_{22}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{1131}}{\partial\phi_{13}\,\partial\phi_{22}} = -1$$

$$\frac{\partial^2 F_{1131}}{\partial\phi_{12}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{1131}}{\partial\phi_{23}\,\partial\phi_{12}} = 1$$

$$\frac{\partial^2 F_{1211}}{\partial\phi_{33}\,\partial\phi_{12}} = -1 \qquad \frac{\partial^2 F_{1211}}{\partial\phi_{12}\,\partial\phi_{33}} = -1 \qquad \frac{\partial^2 F_{1211}}{\partial\phi_{32}\,\partial\phi_{13}} = 1$$

$$\frac{\partial^2 F_{1211}}{\partial\phi_{13}\,\partial\phi_{32}} = 1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{33}\,\partial\phi_{11}} = 1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{33}\,\partial\phi_{22}} = 1$$

$$\frac{\partial^2 F_{1212}}{\partial\phi_{11}\,\partial\phi_{33}} = 1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{22}\,\partial\phi_{33}} = 1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{31}\,\partial\phi_{13}} = -1$$

$$\frac{\partial^2 F_{1212}}{\partial\phi_{13}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{23}\,\partial\phi_{32}} = -1 \qquad \frac{\partial^2 F_{1212}}{\partial\phi_{32}\,\partial\phi_{23}} = -1$$

$$\frac{\partial^2 F_{1213}}{\partial\phi_{11}\,\partial\phi_{32}} = -1 \qquad \frac{\partial^2 F_{1213}}{\partial\phi_{32}\,\partial\phi_{11}} = -1 \qquad \frac{\partial^2 F_{1213}}{\partial\phi_{12}\,\partial\phi_{31}} = 1$$

$$\frac{\partial^2 F_{1213}}{\partial\phi_{31}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{1222}}{\partial\phi_{33}\,\partial\phi_{12}} = -1 \qquad \frac{\partial^2 F_{1222}}{\partial\phi_{12}\,\partial\phi_{33}} = -1$$

$$\frac{\partial^2 F_{1222}}{\partial\phi_{13}\,\partial\phi_{32}} = 1 \qquad \frac{\partial^2 F_{1222}}{\partial\phi_{32}\,\partial\phi_{13}} = 1 \qquad \frac{\partial^2 F_{1232}}{\partial\phi_{22}\,\partial\phi_{13}} = -1$$

$$\frac{\partial^2 F_{1232}}{\partial\phi_{13}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{1232}}{\partial\phi_{12}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{1232}}{\partial\phi_{23}\,\partial\phi_{12}} = 1$$

$$\frac{\partial^2 F_{1311}}{\partial\phi_{22}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{1311}}{\partial\phi_{13}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{1311}}{\partial\phi_{12}\,\partial\phi_{23}} = 1$$

$$\frac{\partial^2 F_{1311}}{\partial\phi_{23}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{1312}}{\partial\phi_{11}\,\partial\phi_{23}} = -1 \qquad \frac{\partial^2 F_{1312}}{\partial\phi_{23}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{1312}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{1312}}{\partial\phi_{21}\,\partial\phi_{13}} = 1 \qquad \frac{\partial^2 F_{1313}}{\partial\phi_{22}\,\partial\phi_{11}} = 1$$

$$
\begin{array}{lll}
\frac{\partial^2 F_{1313}}{\partial\phi_{22}\,\partial\phi_{33}} = 1 & \frac{\partial^2 F_{1313}}{\partial\phi_{11}\,\partial\phi_{22}} = 1 & \frac{\partial^2 F_{1313}}{\partial\phi_{33}\,\partial\phi_{22}} = 1 \\
\frac{\partial^2 F_{1313}}{\partial\phi_{12}\,\partial\phi_{21}} = -1 & \frac{\partial^2 F_{1313}}{\partial\phi_{21}\,\partial\phi_{12}} = -1 & \frac{\partial^2 F_{1313}}{\partial\phi_{23}\,\partial\phi_{32}} = -1 \\
\frac{\partial^2 F_{1313}}{\partial\phi_{32}\,\partial\phi_{23}} = -1 & \frac{\partial^2 F_{1323}}{\partial\phi_{33}\,\partial\phi_{12}} = -1 & \frac{\partial^2 F_{1323}}{\partial\phi_{12}\,\partial\phi_{33}} = -1 \\
\frac{\partial^2 F_{1323}}{\partial\phi_{13}\,\partial\phi_{32}} = 1 & \frac{\partial^2 F_{1323}}{\partial\phi_{32}\,\partial\phi_{13}} = 1 & \frac{\partial^2 F_{1333}}{\partial\phi_{22}\,\partial\phi_{13}} = -1 \\
\frac{\partial^2 F_{1333}}{\partial\phi_{13}\,\partial\phi_{22}} = -1 & \frac{\partial^2 F_{1333}}{\partial\phi_{12}\,\partial\phi_{23}} = 1 & \frac{\partial^2 F_{1333}}{\partial\phi_{23}\,\partial\phi_{12}} = 1 \\
\frac{\partial^2 F_{2111}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 & \frac{\partial^2 F_{2111}}{\partial\phi_{21}\,\partial\phi_{33}} = -1 & \frac{\partial^2 F_{2111}}{\partial\phi_{23}\,\partial\phi_{31}} = 1 \\
\frac{\partial^2 F_{2111}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 & \frac{\partial^2 F_{2121}}{\partial\phi_{33}\,\partial\phi_{11}} = 1 & \frac{\partial^2 F_{2121}}{\partial\phi_{33}\,\partial\phi_{22}} = 1 \\
\frac{\partial^2 F_{2121}}{\partial\phi_{11}\,\partial\phi_{33}} = 1 & \frac{\partial^2 F_{2121}}{\partial\phi_{22}\,\partial\phi_{33}} = 1 & \frac{\partial^2 F_{2121}}{\partial\phi_{13}\,\partial\phi_{31}} = -1 \\
\frac{\partial^2 F_{2121}}{\partial\phi_{31}\,\partial\phi_{13}} = -1 & \frac{\partial^2 F_{2121}}{\partial\phi_{23}\,\partial\phi_{32}} = -1 & \frac{\partial^2 F_{2121}}{\partial\phi_{32}\,\partial\phi_{23}} = -1 \\
\frac{\partial^2 F_{2122}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 & \frac{\partial^2 F_{2122}}{\partial\phi_{21}\,\partial\phi_{33}} = -1 & \frac{\partial^2 F_{2122}}{\partial\phi_{23}\,\partial\phi_{31}} = 1 \\
\frac{\partial^2 F_{2122}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 & \frac{\partial^2 F_{2123}}{\partial\phi_{22}\,\partial\phi_{31}} = -1 & \frac{\partial^2 F_{2123}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \\
\frac{\partial^2 F_{2123}}{\partial\phi_{21}\,\partial\phi_{32}} = 1 & \frac{\partial^2 F_{2123}}{\partial\phi_{32}\,\partial\phi_{21}} = 1 & \frac{\partial^2 F_{2131}}{\partial\phi_{11}\,\partial\phi_{23}} = -1 \\
\frac{\partial^2 F_{2131}}{\partial\phi_{23}\,\partial\phi_{11}} = -1 & \frac{\partial^2 F_{2131}}{\partial\phi_{21}\,\partial\phi_{13}} = 1 & \frac{\partial^2 F_{2131}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \\
\frac{\partial^2 F_{2212}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 & \frac{\partial^2 F_{2212}}{\partial\phi_{21}\,\partial\phi_{33}} = -1 & \frac{\partial^2 F_{2212}}{\partial\phi_{23}\,\partial\phi_{31}} = 1 \\
\frac{\partial^2 F_{2212}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 & \frac{\partial^2 F_{2222}}{\partial\phi_{11}\,\partial\phi_{33}} = 2 & \frac{\partial^2 F_{2222}}{\partial\phi_{33}\,\partial\phi_{11}} = 2 \\
\frac{\partial^2 F_{2222}}{\partial\phi_{13}\,\partial\phi_{31}} = -2 & \frac{\partial^2 F_{2222}}{\partial\phi_{31}\,\partial\phi_{13}} = -2 & \frac{\partial^2 F_{2221}}{\partial\phi_{33}\,\partial\phi_{12}} = -1 \\
\frac{\partial^2 F_{2221}}{\partial\phi_{12}\,\partial\phi_{33}} = -1 & \frac{\partial^2 F_{2221}}{\partial\phi_{13}\,\partial\phi_{32}} = 1 & \frac{\partial^2 F_{2221}}{\partial\phi_{32}\,\partial\phi_{13}} = 1 \\
\frac{\partial^2 F_{2223}}{\partial\phi_{11}\,\partial\phi_{32}} = -1 & \frac{\partial^2 F_{2223}}{\partial\phi_{32}\,\partial\phi_{11}} = -1 & \frac{\partial^2 F_{2223}}{\partial\phi_{12}\,\partial\phi_{31}} = 1
\end{array}
\tag{C.11}
$$

$$\frac{\partial^2 F_{2223}}{\partial\phi_{31}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{2232}}{\partial\phi_{11}\,\partial\phi_{23}} = -1 \qquad \frac{\partial^2 F_{2232}}{\partial\phi_{23}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{2232}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{2232}}{\partial\phi_{21}\,\partial\phi_{13}} = 1$$

$$\frac{\partial^2 F_{2313}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{2313}}{\partial\phi_{21}\,\partial\phi_{33}} = -1 \qquad \frac{\partial^2 F_{2313}}{\partial\phi_{23}\,\partial\phi_{31}} = 1$$

$$\frac{\partial^2 F_{2313}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{2321}}{\partial\phi_{22}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{2321}}{\partial\phi_{13}\,\partial\phi_{22}} = -1$$

$$\frac{\partial^2 F_{2321}}{\partial\phi_{12}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{2321}}{\partial\phi_{23}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{2322}}{\partial\phi_{11}\,\partial\phi_{23}} = -1$$

$$\frac{\partial^2 F_{2322}}{\partial\phi_{23}\,\partial\phi_{11}} = -1 \qquad \frac{\partial^2 F_{2322}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{2322}}{\partial\phi_{21}\,\partial\phi_{13}} = 1$$

$$\frac{\partial^2 F_{2323}}{\partial\phi_{11}\,\partial\phi_{22}} = 1 \qquad \frac{\partial^2 F_{2323}}{\partial\phi_{11}\,\partial\phi_{33}} = 1 \qquad \frac{\partial^2 F_{2323}}{\partial\phi_{22}\,\partial\phi_{11}} = 1$$

$$\frac{\partial^2 F_{2323}}{\partial\phi_{33}\,\partial\phi_{11}} = 1 \qquad \frac{\partial^2 F_{2323}}{\partial\phi_{12}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{2323}}{\partial\phi_{21}\,\partial\phi_{12}} = -1$$

$$\frac{\partial^2 F_{2323}}{\partial\phi_{13}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{2323}}{\partial\phi_{31}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{2333}}{\partial\phi_{11}\,\partial\phi_{23}} = -1$$

$$\frac{\partial^2 F_{2333}}{\partial\phi_{23}\,\partial\phi_{11}} = -1 \qquad \frac{\partial^2 F_{2333}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{2333}}{\partial\phi_{21}\,\partial\phi_{13}} = 1$$

$$\frac{\partial^2 F_{3111}}{\partial\phi_{22}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{3111}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{3111}}{\partial\phi_{21}\,\partial\phi_{32}} = 1$$

$$\frac{\partial^2 F_{3111}}{\partial\phi_{32}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{3121}}{\partial\phi_{11}\,\partial\phi_{32}} = -1 \qquad \frac{\partial^2 F_{3121}}{\partial\phi_{32}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{3121}}{\partial\phi_{12}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{3121}}{\partial\phi_{31}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{3131}}{\partial\phi_{22}\,\partial\phi_{11}} = 1$$

$$\frac{\partial^2 F_{3131}}{\partial\phi_{22}\,\partial\phi_{33}} = 1 \qquad \frac{\partial^2 F_{3131}}{\partial\phi_{11}\,\partial\phi_{22}} = 1 \qquad \frac{\partial^2 F_{3131}}{\partial\phi_{33}\,\partial\phi_{22}} = 1$$

$$\frac{\partial^2 F_{3131}}{\partial\phi_{12}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{3131}}{\partial\phi_{21}\,\partial\phi_{12}} = -1 \qquad \frac{\partial^2 F_{3131}}{\partial\phi_{23}\,\partial\phi_{32}} = -1$$

$$\frac{\partial^2 F_{3131}}{\partial\phi_{32}\,\partial\phi_{23}} = -1 \qquad \frac{\partial^2 F_{3132}}{\partial\phi_{33}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{3132}}{\partial\phi_{21}\,\partial\phi_{33}} = -1$$

$$\frac{\partial^2 F_{3132}}{\partial\phi_{23}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{3132}}{\partial\phi_{31}\,\partial\phi_{23}} = 1 \qquad \frac{\partial^2 F_{3133}}{\partial\phi_{22}\,\partial\phi_{31}} = -1$$

$$\frac{\partial^2 F_{3133}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{3133}}{\partial\phi_{21}\,\partial\phi_{32}} = 1 \qquad \frac{\partial^2 F_{3133}}{\partial\phi_{32}\,\partial\phi_{21}} = 1$$

$$\frac{\partial^2 F_{3212}}{\partial\phi_{22}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{3212}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{3212}}{\partial\phi_{21}\,\partial\phi_{32}} = 1$$

$$\frac{\partial^2 F_{3212}}{\partial\phi_{32}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{3222}}{\partial\phi_{11}\,\partial\phi_{32}} = -1 \qquad \frac{\partial^2 F_{3222}}{\partial\phi_{32}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{3222}}{\partial\phi_{12}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{3222}}{\partial\phi_{31}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{3231}}{\partial\phi_{33}\,\partial\phi_{12}} = -1$$

$$\frac{\partial^2 F_{3231}}{\partial\phi_{12}\,\partial\phi_{33}} = -1 \qquad \frac{\partial^2 F_{3231}}{\partial\phi_{13}\,\partial\phi_{32}} = 1 \qquad \frac{\partial^2 F_{3231}}{\partial\phi_{32}\,\partial\phi_{13}} = 1$$

$$\frac{\partial^2 F_{3232}}{\partial\phi_{11}\,\partial\phi_{22}} = 1 \qquad \frac{\partial^2 F_{3232}}{\partial\phi_{11}\,\partial\phi_{33}} = 1 \qquad \frac{\partial^2 F_{3232}}{\partial\phi_{22}\,\partial\phi_{11}} = 1$$

$$\frac{\partial^2 F_{3232}}{\partial\phi_{33}\,\partial\phi_{11}} = 1 \qquad \frac{\partial^2 F_{3232}}{\partial\phi_{12}\,\partial\phi_{21}} = -1 \qquad \frac{\partial^2 F_{3232}}{\partial\phi_{21}\,\partial\phi_{12}} = -1$$

$$\frac{\partial^2 F_{3232}}{\partial\phi_{13}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{3232}}{\partial\phi_{31}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{3233}}{\partial\phi_{11}\,\partial\phi_{32}} = -1$$

$$\frac{\partial^2 F_{3233}}{\partial\phi_{32}\,\partial\phi_{11}} = -1 \qquad \frac{\partial^2 F_{3233}}{\partial\phi_{12}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{3233}}{\partial\phi_{31}\,\partial\phi_{12}} = 1$$

$$\frac{\partial^2 F_{3313}}{\partial\phi_{22}\,\partial\phi_{31}} = -1 \qquad \frac{\partial^2 F_{3313}}{\partial\phi_{31}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{3313}}{\partial\phi_{21}\,\partial\phi_{32}} = 1$$

$$\frac{\partial^2 F_{3313}}{\partial\phi_{32}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{3323}}{\partial\phi_{11}\,\partial\phi_{32}} = -1 \qquad \frac{\partial^2 F_{3323}}{\partial\phi_{32}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{3323}}{\partial\phi_{12}\,\partial\phi_{31}} = 1 \qquad \frac{\partial^2 F_{3323}}{\partial\phi_{31}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{3333}}{\partial\phi_{11}\,\partial\phi_{22}} = 2$$

$$\frac{\partial^2 F_{3333}}{\partial\phi_{22}\,\partial\phi_{11}} = 2 \qquad \frac{\partial^2 F_{3333}}{\partial\phi_{12}\,\partial\phi_{21}} = -2 \qquad \frac{\partial^2 F_{3333}}{\partial\phi_{21}\,\partial\phi_{12}} = -2$$

$$\frac{\partial^2 F_{3331}}{\partial\phi_{22}\,\partial\phi_{13}} = -1 \qquad \frac{\partial^2 F_{3331}}{\partial\phi_{13}\,\partial\phi_{22}} = -1 \qquad \frac{\partial^2 F_{3331}}{\partial\phi_{12}\,\partial\phi_{23}} = 1$$

$$\frac{\partial^2 F_{3331}}{\partial\phi_{23}\,\partial\phi_{12}} = 1 \qquad \frac{\partial^2 F_{3332}}{\partial\phi_{11}\,\partial\phi_{23}} = -1 \qquad \frac{\partial^2 F_{3332}}{\partial\phi_{23}\,\partial\phi_{11}} = -1$$

$$\frac{\partial^2 F_{3332}}{\partial\phi_{13}\,\partial\phi_{21}} = 1 \qquad \frac{\partial^2 F_{3332}}{\partial\phi_{21}\,\partial\phi_{13}} = 1$$

Appendix D

Numerical Simulation Code

The numerical simulation of high cycle fatigue damage is performed using a FORTRAN 77 code. The program, called *MMF* (Metal Matrix Fatigue), reads the input data and outputs the input information into a file for verification purposes. The results from the numerical high cycle fatigue simulation are written to output files from which the necessary information may be extracted.

A typical input file: MEMAFA.IN

```
1   ----- Output file for Phi data:
2   dam_f_m.dat
3   ----- Comments (5 lines only)  regarding the equation of Xi_m
4      Xi_1 = parabolic (see maple file) such that Xi_1(0) = Xi0
5           and Xi_1(N1) = Xi1 and dXi_1(N1)/dN = dXi_2/dN|(N1)
6      Xi_2 = exponential such that Xi_2(N1) = Xi1 and Xi_2(N2) = Xi2
7      Xi_3 = parabolic with Xi_3(N2) = Xi2 and dXi_3/dN(N2) = dXi_2/dN(N2);
8           the remaining coefficient a controls the rate of failure
9   ----- Comments (5 lines only)  regarding the equation of Xi_f
10     Xi_1 = parabolic (see maple file) such that Xi_1(0) = Xi0
11          and Xi_1(N1) = Xi1 and dXi_1(N1)/dN = dXi_2/dN|(N1)
12     Xi_2 = exponential such that Xi_2(N1) = Xi1 and Xi_2(N2) = Xi2
13     Xi_3 = parabolic with Xi_3(N2) = Xi2 and dXi_3/dN(N2) = dXi_2/dN(N2);
14          the remaining coefficient a controls the rate of failure
15  ----- Flag to indicate whether to write after each cycle the results or
16  ----- dump them in one shot: 0 = dump 1 = write
17  0
18  ----- Miscellaneous Flag
19  0
20  ----- Flag to determine whether to adjust concentration tensors:
21  ----- 0=no adjustment;1=adjustment
22  1
23  ----- Flag to determine whether to use effective volume fractions or original:
24  ----- 0 = use original; 1 = use effective
25  0
26  ----- Load_Flag: 1 = Sinusoidal | 2 = Triangular | 4 = Monotonic
27  1
28  ----- Flag to indicate whether current run is restart or new:
29  ----- 0 = new ; 1 = restart
```

```
0
----- Flag to indicate whether to include interface damage:
----- 0 = do not include; 1 = do include
0
----- Number of Load Cycles
30
----- Mean_Stress (MPa)
 550.    0.    0.
   0.    0.    0.
   0.    0.    0.
----- Stress_Amplitude (MPa)
 450.    0.    0.
   0.    0.    0.
   0.    0.    0.
----- Number_of_Discretization_points_for_curve
25
----- Stepsize in case of damage for the mean stress part (in MPa)
1.0
----- Stepsize in case of damage for the cyclic part (in degrees)
0.2
----- Material_Flag
1
----- Fiber (MPa)
0.325
0.25
4.0d5
0.
0.
0.
0.
0.
0.
0.
----- Matrix (MPa)
0.36
9.180d4
0.
0.
0.
0.
0.
0.
0.
----- Interface (MPa)
0.3
1.d4
0.
0.
0.
0.
0.
```

```
0.
0.
=============== Damage_Parameters:_Matrix
----- Power of F in g:
1.d0
----- Coefficient c_Gamma
1.d0
----- Kappa_0
1.d-20
----- Lambda (MPa)
8.6786d4    8.6786d4    8.6786d4
---- Eta
1.d0   1.d0   1.d0
----- Xi_0
0.70d0    0.70d0    0.70d0
----- dXi_m1
3.d-2     3.d-2     3.d-2
----- dXi_m2
6.d-2     6.d-2     6.d-2
----- Parameter a_3
0.5d0
----- Damage_Threshold Coefficient Vi (MPa)
5.d-2    5.d-2    5.d-2
----- Critical Damage Parameter:
.40d0    .00d0   .00d0
.00d0    .40d0   .00d0
.00d0    .00d0   .40d0
----- Limits of Cycles (N_m1,N_m2)
10
50000
=============== Damage_Parameters:_Fiber
----- Power of F in g:
1.d0
----- Coefficient c_Gamma
1.d0
----- Kappa_0
1.d-20
----- Lambda (MPa)
1.60d5   1.60d5   1.60d5
----- Eta
1.d0    1.d0   1.d0
----- Xi_0
.65d0   .65d0  .65d0
----- dXi_f1
3.d-2    3.d-2    3.d-2
----- dXi_f2
5.0d-2    5.0d-2    5.0d-2
----- Parameter a_3
0.25d0
----- Damage_Threshold Coefficient Vi (MPa)
3.d0    3.d0    3.d0
```

```
132  ----- Critical Damage Parameter:
133  .40d0     .00d0    .00d0
134  .00d0     .40d0    .00d0
135  .00d0     .00d0    .40d0
136  ----- Limits of Cycles (N_f1,N_f2)
137  10
138  70000
139  ============== Damage_Parameters:_Interface
140  ----- Power of F in g:
141  1.d0
142  ----- Coefficient c_Gamma
143  1.d0
144  ----- Kappa_0
145  1.d-20
146  ----- Lambda (MPa)
147  9.d4    9.d4    9.d4
148  ----- Eta
149  1.d0     1.d0    1.d0
150  ----- Xi_0
151  .5d0    .5d0   .5d0
152  ----- dXi_f1
153  .2d0   .2d0   .2d0
154  ----- dXi_f2
155  0.1d0   0.1d0   0.1d0
156  ----- Parameter a_3
157  0.d0
158  ----- Damage_Threshold Coefficient Vi (MPa)
159  5.7d0    5.7d0   5.7d0
160  ----- Critical Damage Parameter:
161  .25d0     .25d0    .25d0
162  .25d0     .25d0    .25d0
163  .25d0     .25d0    .25d0
164  ----- Limits of Cycles (N_i1,N_i2)
165  10
166  10000
```

Verification of input: MEMAFA.OUT

```
Equation of Xi_m
   Xi_1 = parabolic (see maple file) such that Xi_1(0) = Xi0
        and Xi_1(N1) = Xi1 and dXi_1(N1)/dN = dXi_2/dN|(N1)
   Xi_2 = exponential such that Xi_2(N1) = Xi1 and Xi_2(N2) = Xi2
   Xi_3 = parabolic with Xi_3(N2) = Xi2 and dXi_3/dN(N2) = dXi_2/dN(N2);
        the remaining coefficient a controls the rate of failure
Equation of Xi_f
   Xi_1 = parabolic (see maple file) such that Xi_1(0) = Xi0
        and Xi_1(N1) = Xi1 and dXi_1(N1)/dN = dXi_2/dN|(N1)
   Xi_2 = exponential such that Xi_2(N1) = Xi1 and Xi_2(N2) = Xi2


   Xi_3 = parabolic with Xi_3(N2) = Xi2 and dXi_3/dN(N2) = dXi_2/dN(N2);
        the remaining coefficient a controls the rate of failure

-------------------------------------------------------------
Input Data
==========

---> Initial volume fractions used !!

---> Effective concentration tensors used !!

Loading Parameters
------------------
   Mean Stress (MPa):

    550.0000         0.0000         0.0000
      0.0000         0.0000         0.0000
      0.0000         0.0000         0.0000


   Stress Amplitude (MPa):

    450.0000         0.0000         0.0000
      0.0000         0.0000         0.0000
      0.0000         0.0000         0.0000

   Number of Discretization Points (per Curve):                    25
   Stress increment in case of damage (mean stress, MPa):  0.100000D+01
   Increment in case of damage (cyclic range, in degrees): 0.200000D+00


Isotropic Material Data
-----------------------

  Fiber:
   c_f_0 =         0.325
   Nu_12 =         0.250
```

```
50      E_1   =    400000.000
51      G_12  =         0.000
52      Nu_13 =         0.000
53      E_2   =         0.000
54      G_13  =         0.000
55      Nu_23 =         0.000
56      E_3   =         0.000
57      G_23  =         0.000
58
59     Matrix:
60      c_m   =         0.675
61      Nu_12 =         0.360
62      E_1   =     91800.000
63      G_12  =         0.000
64      Nu_13 =         0.000
65      E_2   =         0.000
66      G_13  =         0.000
67      Nu_23 =         0.000
68      E_3   =         0.000
69      G_23  =         0.000
70
71     Interface:
72      Nu_12 =         0.300
73      E_1   =     10000.000
74      G_12  =         0.000
75      Nu_13 =         0.000
76      E_2   =         0.000
77      G_13  =         0.000
78      Nu_23 =         0.000
79      E_3   =         0.000
80      G_23  =         0.000
81
82   Damage Parameters
83   -----------------
84     Matrix:
85
86       Power of F  =  0.1000D+01
87       c_Gamma_m   =  0.1000D+01
88       Kappa_m_0   =  0.1000D-19
89       Lambda(1,1) =  0.8679D+05
90       Lambda(2,2) =  0.8679D+05
91       Lambda(3,3) =  0.8679D+05
92       Eta(1,1)    =  0.1000D+01
93       Eta(2,2)    =  0.1000D+01
94       Eta(3,3)    =  0.1000D+01
95       Xi(1,1)     =  0.7000D+00
96       Xi(2,2)     =  0.7000D+00
97       Xi(3,3)     =  0.7000D+00
98       dXi_1(1,1)  =  0.3000D-01
99       dXi_1(2,2)  =  0.3000D-01
100      dXi_1(3,3)  =  0.3000D-01
```

```
 101     dXi_2(1,1)  =  0.6000D-01
 102     dXi_2(2,2)  =  0.6000D-01
 103     dXi_2(3,3)  =  0.6000D-01
 104     a_m         =  0.5000D+00
 105     Vi(1,1)     =  0.5000D-01
 106     Vi(2,2)     =  0.5000D-01
 107     Vi(3,3)     =  0.5000D-01
 108
 109    - Critical Damage Parameters:
 110
 111     0.4000       0.0000       0.0000
 112     0.0000       0.4000       0.0000
 113     0.0000       0.0000       0.4000
 114
 115    - Limit Values for Cycles
 116
 117     Lower Limit =     10
 118     Upper Limit =  50000
 119
 120   Fiber:
 121
 122     Power of F  =  0.1000D+01
 123     c_Gamma_f   =  0.1000D+01
 124     Kappa_f_0   =  0.1000D-19
 125     Lambda(1,1) =  0.1600D+06
 126     Lambda(2,2) =  0.1600D+06
 127     Lambda(3,3) =  0.1600D+06
 128     Eta(1,1)    =  0.1000D+01
 129     Eta(2,2)    =  0.1000D+01
 130     Eta(3,3)    =  0.1000D+01
 131     Xi(1,1)     =  0.6500D+00
 132     Xi(2,2)     =  0.6500D+00
 133     Xi(3,3)     =  0.6500D+00
 134     dXi_1(1,1)  =  0.3000D-01
 135     dXi_1(2,2)  =  0.3000D-01
 136     dXi_1(3,3)  =  0.3000D-01
 137     dXi_2(1,1)  =  0.5000D-01
 138     dXi_2(2,2)  =  0.5000D-01
 139     dXi_2(3,3)  =  0.5000D-01
 140     a_f         =  0.2500D+00
 141     Vi(1,1)     =  0.3000D+01
 142     Vi(2,2)     =  0.3000D+01
 143     Vi(3,3)     =  0.3000D+01
 144
 145    - Critical Damage Parameters:
 146
 147     0.4000       0.0000       0.0000
 148     0.0000       0.4000       0.0000
 149     0.0000       0.0000       0.4000
 150
 151    - Limit Values for Cycles
```

```
152
153     Lower Limit =     10
154     Upper Limit =  70000
155
156   Interface:
157
158     Power of F  =  0.1000D+01
159     c_Gamma_i   =  0.1000D+01
160     Kappa_i_0   =  0.1000D-19
161     Lambda(1,1) =  0.9000D+05
162     Lambda(2,2) =  0.1600D+06
163     Lambda(3,3) =  0.1600D+06
164     Eta(1,1)    =  0.1000D+01
165     Eta(2,2)    =  0.1000D+01
166     Eta(3,3)    =  0.1000D+01
167     Xi(1,1)     =  0.5000D+00
168     Xi(2,2)     =  0.5000D+00
169     Xi(3,3)     =  0.5000D+00
170     dXi_1(1,1)  =  0.2000D+00
171     dXi_1(2,2)  =  0.2000D+00
172     dXi_1(3,3)  =  0.2000D+00
173     dXi_2(1,1)  =  0.1000D+00
174     dXi_2(2,2)  =  0.1000D+00
175     dXi_2(3,3)  =  0.1000D+00
176     a_i         =  0.0000D+00
177     Vi(1,1)     =  0.5700D+01
178     Vi(2,2)     =  0.5700D+01
179     Vi(3,3)     =  0.5700D+01
180
181   - Critical Damage Parameters:
182
183     0.2500       0.2500       0.2500
184     0.2500       0.2500       0.2500
185     0.2500       0.2500       0.2500
186
187   - Limit Values for Cycles
188
189     Lower Limit =     10
190     Upper Limit =  10000
```

Output data from simulation: DAM_F_M.DAT

1	Cycle	Phi_m(1,1)	Sigma_m(1,1)	Xi_m(1,1)	Phi_f(1,1)	Sigma_f(1,1)	Xi_f(1,1)	Phi_c(1,1)	Sigma_tot(1,1)
2	0	0.00000000D+00	0.00000000D+00	0.00000000D+00	0.00000000D+00	0.00000000D+00	0.00000000D+00	0.00000000D+00	0.00000000D+00
3	1	0.30939343D-01	0.27932276D+03	0.70000000D+00	0.56465296D-02	0.11608382D+04	0.65000000D+00	0.22719178D-01	0.55000000D+03
4	2	0.35138903D-01	0.28121336D+03	0.71206323D+00	0.70692556D-02	0.11652983D+04	0.66260713D+00	0.26016268D-01	0.55000000D+03
5	3	0.37481300D-01	0.28227708D+03	0.71790896D+00	0.78635305D-02	0.11678007D+04	0.66855385D+00	0.27855525D-01	0.55000000D+03
6	4	0.39049135D-01	0.28299124D+03	0.72151455D+00	0.83874209D-02	0.11694659D+04	0.67213387D+00	0.29084078D-01	0.55000000D+03
7	5	0.40193372D-01	0.28351294D+03	0.72400143D+00	0.87625818D-02	0.11706697D+04	0.67454533D+00	0.29978365D-01	0.55000000D+03
8	6	0.41071423D-01	0.28391317D+03	0.72583242D+00	0.90443393D-02	0.11716830D+04	0.67627866D+00	0.30662621D-01	0.55000000D+03
9	7	0.41767112D-01	0.28422987D+03	0.72723951D+00	0.92620052D-02	0.11722965D+04	0.67757790D+00	0.31202952D-01	0.55000000D+03
10	8	0.42330501D-01	0.28448606D+03	0.72835397D+00	0.94345860D-02	0.11728676D+04	0.67858020D+00	0.31639329D-01	0.55000000D+03
11	9	0.42797150D-01	0.28469769D+03	0.72925653D+00	0.95730413D-02	0.11733322D+04	0.67936940D+00	0.31999316D-01	0.55000000D+03
12	10	0.43188637D-01	0.28487473D+03	0.73000000D+00	0.96855364D-02	0.11737150D+04	0.68000000D+00	0.32300129D-01	0.55000000D+03
13	11	0.43532605D-01	0.28503014D+03	0.73064554D+00	0.97828425D-02	0.11740485D+04	0.68053825D+00	0.32563932D-01	0.55000000D+03
14	12	0.43849836D-01	0.28517358D+03	0.73123536D+00	0.98725056D-02	0.11743561D+04	0.68102964D+00	0.32807204D-01	0.55000000D+03
15	13	0.44146468D-01	0.28530774D+03	0.73177837D+00	0.99560029D-02	0.11746432D+04	0.68148167D+00	0.33034567D-01	0.55000000D+03
16	14	0.44425671D-01	0.28543398D+03	0.73228148D+00	0.10033872D-01	0.11749121D+04	0.68190019D+00	0.33248336D-01	0.55000000D+03
17	15	0.44687242D-01	0.28555234D+03	0.73276017D+00	0.10106825D-01	0.11751642D+04	0.68228981D+00	0.33448607D-01	0.55000000D+03
18	16	0.44934503D-01	0.28566441D+03	0.73318888D+00	0.10176412D-01	0.11754037D+04	0.68265429D+00	0.33638124D-01	0.55000000D+03
19	17	0.45167316D-01	0.28576992D+03	0.73360122D+00	0.10241542D-01	0.11756286D+04	0.68299666D+00	0.33816439D-01	0.55000000D+03
20	18	0.45390492D-01	0.28587110D+03	0.73399019D+00	0.10303787D-01	0.11758439D+04	0.68331945D+00	0.33987313D-01	0.55000000D+03
21	19	0.45603428D-01	0.28596774D+03	0.73435832D+00	0.10363478D-01	0.11760500D+04	0.68362479D+00	0.34150445D-01	0.55000000D+03
22	20	0.45807091D-01	0.28606006D+03	0.73470773D+00	0.10419706D-01	0.11762455D+04	0.68391446D+00	0.34306191D-01	0.55000000D+03
23	21	0.46002390D-01	0.28614869D+03	0.73504025D+00	0.10473931D-01	0.11764336D+04	0.68419000D+00	0.34455641D-01	0.55000000D+03
24	22	0.46189990D-01	0.28623394D+03	0.73535743D+00	0.10526347D-01	0.11766150D+04	0.68445272D+00	0.34599306D-01	0.55000000D+03
25	23	0.46370498D-01	0.28631691D+03	0.73566064D+00	0.10576332D-01	0.11767887D+04	0.68470375D+00	0.34737394D-01	0.55000000D+03
26	24	0.46544601D-01	0.28639493D+03	0.73595106D+00	0.10624075D-01	0.11769553D+04	0.68494410D+00	0.34870430D-01	0.55000000D+03
27	25	0.46712622D-01	0.28647128D+03	0.73622974D+00	0.10670483D-01	0.11771169D+04	0.68517464D+00	0.34998927D-01	0.55000000D+03
28	26	0.46875148D-01	0.28654516D+03	0.73649757D+00	0.10715322D-01	0.11772731D+04	0.68539613D+00	0.35123205D-01	0.55000000D+03
29	27	0.47032417D-01	0.28661674D+03	0.73675540D+00	0.10759069D-01	0.11774249D+04	0.68560927D+00	0.35243579D-01	0.55000000D+03
30	28	0.47184883D-01	0.28668608D+03	0.73700393D+00	0.10801038D-01	0.11775714D+04	0.68581465D+00	0.35360134D-01	0.55000000D+03
31	29	0.47332907D-01	0.28675343D+03	0.73724381D+00	0.10841738D-01	0.11777134D+04	0.68601282D+00	0.35473277D-01	0.55000000D+03
32	30	0.47476660D-01	0.28681876D+03	0.73747564D+00	0.10880812D-01	0.11778506D+04	0.68620428D+00	0.35583009D-01	0.55000000D+03

Main program source code: FATIGUE.F

```
C ------------------------------------------------------------------
      PROGRAM MMF
C ------------------------------------------------------------------
C     Metal Matrix Fatigue
C ------------------------------------------------------------------
C     Program to analyze the fatigue behavior and life time of a Metal
C     Matrix Composite (MMC) material. It is required to provide the
C     following input data: (see input data file and/or read statements
C     for exact number of variables)
C ------------------------------------------------------------------
C     Description of Variables:
C
C     A_f,A_m           4th order tensors representing the strain concentrati
C                       tensors for the fiber and matrix
C     A_m_bar           effective concentration tensors
C     A_f_bar
C
C     B_f,B_m           4th order tensors representing the stress concentrati
C                       tensors for the fiber and matrix
C     B_m_bar           effective concentration tensors
C     B_f_bar
C
C     c_f_0             Volume fraction of the fibers for the virgin material
C     c_m_0             Volume fraction of the matrix for the virgin material
C
C     Damage_Stepsize   Factor to stretch or reduce the standard load increme
C                       of 1.0 for the damage region
C     Delta             Kronecker DELTA
C     Delta_Sigma       cycling amplitude
C     DDM               8th order tensor representing the 2nd partial
C                       derivative of M_rsuv wrt Phi_ij Phi_kl
C     D2M               8th order tensor representing the 2nd partial
C                       derivative of M_rsuv wrt Phi_ij Phi_kl stored
C                       in a 1-d array
C     DM                6th order tensor representing the 1st partial
C                       derivative of M_rsuv wrt Phi_ij
C
C     dPhi              damage increment
C     dPhi_adj          adjusted damage increment
C
C     dSigma_mean       mean stress increment
C     dSigma_tot        adaptive total stress increment
C     dSigma_tot_0      initial fixed total stress increment
C     dSigma_f          stress increment in the fiber
C     dSigma_m          stress increment in the matrix
C     dSigma_i          stress increment in the interface
C
C     E_f_bar           4th order undamaged Elasticity tensor for the
C                       fiber material
C     E_m_bar           4th order undamaged Elasticity tensor for the
C                       matrix material
C     E_bar             4th order undamaged Elasticity tensor (global)
C     Eta_f             vector containing material varibale for the fiber
C     Eta_m             vector containing material varibale for the matrix
C     Eta               vector containing material varibale (global)
C
C     ID                integer variable to identify the material
C     IFlag_Damage      Flag which indicates whether damage occured in the
C                       current increment. It is used to reduce the stress
C                       increment to the fixed value for the damage region
C     IFlag_damage_constituent    Integer array which is used to store the
C                                 flags to indicate in which constituent
C                                 damage occurred
C     IFlag_EVF         effective volume fraction flag
C                       0 = use original volume fractions
C                       1 = use effective volume fractions
C     IFlag_effstress         flag to indicate whether the effective stress
C                       concentration tensors will be used
C                       = 0       regular concentration tensors is used
C                       = 1       effective concentration tensor is used
C     IFlag_Restart     flag to indicate whether the current run is a restart
C                       of a previous completed run
C                       = 0   ---> new run
C                       = 1   ---> restart
C     IFlag_min_step    Flag to indicate whether the minimum stepsize
C                       requirement controls the stepsize chosen
C                       0 = stepsize larger than minimum stepsize used
C                       1 = minimum stepsize used
```

```
C     IFlag_step        Flag to indicate whether the stepsize chosen is ok
C                       0 = not acceptable
C                       1 = acceptable
C     IFlag_stepsize    Flag array to indicate whether the stepsize chosen is
C                       acceptable for the constituents.
C                       0 = not acceptable
C                       1 = acceptable
C
C     Kappa_f           Damage hardening variable for the fiber
C     Kappa_m           Damage hardening variable for the matrix
C     Kappa             Damage hardening variable used during calculation
C
C     Lambda_f          vector containing the material parameter for the matri
C     Lambda_m          vector containing the material parameter for the fiber
C     Lambda            vector containing the material parameter (global)
C     L_inc             number Of load increments
C
C     M_f               4th oder damage effect tensor for the fiber
C     M_m               4th oder damage effect tensor for the matrix
C     M_comp            4th oder damage effect tensor for the composite
C     M                 4th oder damage effect tensor used globally
C
C     N_inc_mean        number of load increments to the mean stress; this
C                       value is going to be used to calculate dSigma_tot_0
C     N_output          device number where regular output is written to
C     Number_of_cycles  total number of cycles to be applied
C     Nu_f              2nd order tensor for Poisson's ratios of the fiber
C     Nu_m              2nd order tensor for Poisson's ratios of the matrix
C
C     Phi_f,Phi_m       2nd order tensor representing the damage in the fiber
C                       matrix material respectively
C     Phi_old,          2nd order tensors representing the damage at the
C     Phi_new           current state and new state, respectively
C
C     Sigma_f           2nd order tensor containing the state of stress in
C                       the fiber material
C     Sigma_m           2nd order tensor containing the state of stress in
C                       the matrix material
C     Sigma_old         2nd order tensor containing the state of stress at th
C                       current state
C     Sigma_nold        2nd order tensor containing the state of stress at th
C                       current state if stress subincremented
C
C     Sigma_new         2nd order tensor containing the state of stress at the
C                       new state
C     Sigma_tot         2nd order tensor containing the total state of stress
C     Sigma_tot_eff     2nd order tensor containing the effective total state
C                       of stress acting on the composite
C     Step_factor       Factor which specifies by how much the stress increment
C                       is going to be reduced if a stress increment adjustment
C                       has to be made
C
C     TOL_delta_g       Value which specifies how much to subsequent g values
C                       may differ such that the appropriate stress increment
C                       is acceptable
C
C     V_f,V_m           2nd order tensor containing the damage threshold for
C                       fiber/matrix material respectively
C     V                 2nd order tensor containing the damage threshold
C                       (global)
C
C     w_old             variable for the current state of stress
C
C     Xi_f              vector containing the material variable for the fiber
C     Xi_m              vector containing the material variable for the matrix
C     Xi                vector containing the material variable (global)
C-----------------------------------------------------------------------
C     Author:     Rainer Echle
C                 Department of Civil Engineering
C                 Louisiana State University
C                 Baton Rouge LA 70803
C
C     Written:    07-11-1994
C     Updated:    12-20-1996
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      REAL*8    Lambda(3),M(3,3,3,3),Kappa_old,Kappa_new,I4(3,3,3,3)
C
      DIMENSION Delta(3,3),
     +          dF_pot_dY(3,3),dg_dY(3,3),dg_dSigma(3,3),dg_dPhi(3,3),
     +          dg_dGamma(3,3),
     +          dY_dSigma(3,3,3,3),dGamma_dPhi(3,3,3,3),
     +          dF_pot_dGamma(3,3),dY_dPhi(3,3,3,3),dF_pot_dw(3,3),
     +          dw_dKappa(3,3),dF_pot_dSigma(3,3),
```

```
      +          dum2_1(3,3),dum2_2(3,3),dum2_3(3,3),dum2_4(3,3),
      +          Prop(9),
      +          w_old(3,3),Eta(3),Xi(3),V(3),dXi(3),
      +          w_inv(3,3),w_new(3,3),
      +          Phi_old(3,3),Phi_new(3,3),dPhi(3,3),dPhi_adj(3,3),
      +          Phi_dum(3,3),
      +          Y_old(3,3),Y_new(3,3),dY(3,3),
      +          Gamma_old(3,3),Gamma_new(3,3),dGamma(3,3),
      +          DM(3,3,3,3,3,3),D2M(6561),
      +          E_bar(3,3,3,3),E_bar_inv(3,3,3,3),S(3,3,3,3),
      +          Psi_num(3,3,3,3),Psi(3,3,3,3),
      +          IFlag_stepsize(3),
      +          IFlag_damage_constituent(3),
      +          Results_damage(500000,9),EVF(500000,2)
C
      CHARACTER*80 BUFFER
      CHARACTER*132 Comment(10)
      CHARACTER*2  IMP
C -----------------------------------------------------------------------
      include 'common.f'
      include 'devices.f'
C -----------------------------------------------------------------------
C
C     Initialize the variables to zero
C
      CALL DINITIALIZE_ZERO_4(E_f_bar,3)
      CALL DINITIALIZE_ZERO_4(E_m_bar,3)
      CALL DINITIALIZE_ZERO_2(Nu_f,3)
      CALL DINITIALIZE_ZERO_2(Nu_m,3)
      CALL DINITIALIZE_ZERO_2(Sigma_Mean,3)
      CALL DINITIALIZE_ZERO_2(Sigma_tot,3)
      CALL DINITIALIZE_ZERO_2(Delta_Sigma,3)
      CALL DINITIALIZE_ZERO_2(dSigma,3)
      Theta_old = 0.D0
      CALL DINITIALIZE_DELTA(Delta)
C -----------------------------------------------------------------------
C     Constants
C -----------------------------------------------------------------------
      Pi                = DATAN(1.D0)*4
      TOL_delta_g       = 0.1d0
      Damage_Stepsize   = 1.d0
      Step_factor       = 0.5d0
      Iflag_damage      = 0
C
      Kappa_f_0 = 1.d-20
      Kappa_m_0 = 1.d-20
      Kappa_i_0 = 1.d-20
C -----------------------------------------------------------------------
C     Assign UNIT numbers for any kind of input and output
C -----------------------------------------------------------------------
      Mori      = N_misc
      Esh       = N_misc
      Numlib    = N_misc
      Tensinv   = N_misc
      Yij       = N_misc
C -----------------------------------------------------------------------
C     Assign file names to the UNITS
C -----------------------------------------------------------------------
      F_Error      = 'error.dat'
      File_m       = 'check_m.dat'
      File_f       = 'check_f.dat'
      File_r       = 'check_r.dat'
      File_i       = 'check_i.dat'
      File_i1      = 'check_i1.dat'
      File_i2      = 'check_i2.dat'
      File_Restart = 'restart.dat'
      F_misc       = 'misc.dat'
C
      THETA_OLD = 0.D0
C -----------------------------------------------------------------------
      OPEN(7,FILE='fatitest.in',STATUS='OLD')
C -----------------------------------------------------------------------
C     Determine the input file
C -----------------------------------------------------------------------
      READ(7,'(A)') BUFFER
      ID = INDEX(BUFFER,'.')
      FILE_1=BUFFER(1:ID+3)
C -----------------------------------------------------------------------
C     Determine the output file
C -----------------------------------------------------------------------
      READ(7,'(A)') BUFFER
      ID = INDEX(BUFFER,'.')
      FILE_2=BUFFER(1:ID+3)
C -----------------------------------------------------------------------
```

```
      OPEN(N_Input,FILE=FILE_1,STATUS='UNKNOWN')
      OPEN(N_Output,FILE=FILE_2,STATUS='UNKNOWN')
      OPEN(N_Error,FILE=F_Error,STATUS='UNKNOWN')
      OPEN(Check_r,FILE=File_r,STATUS='UNKNOWN')
      OPEN(Check_i1,FILE=File_i1,STATUS='UNKNOWN')
      OPEN(Check_i2,FILE=File_i2,STATUS='UNKNOWN')
      OPEN(N_Restart,FILE=File_Restart,STATUS='UNKNOWN')
C
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      ID = INDEX(BUFFER,'.')
      FILE_2=BUFFER(1:ID+3)
C
      IF (N_misc.gt.0) OPEN(N_misc,FILE=F_misc,STATUS='UNKNOWN')
C -----------------------------------------------------------------------
C     Read all the necessary input data
C     (for monotonic loading right now)
C
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Comment(1)
      READ(N_input,'(A)') Comment(2)
      READ(N_input,'(A)') Comment(3)
      READ(N_input,'(A)') Comment(4)
      READ(N_input,'(A)') Comment(5)
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Comment(6)
      READ(N_input,'(A)') Comment(7)
      READ(N_input,'(A)') Comment(8)
      READ(N_input,'(A)') Comment(9)
      READ(N_input,'(A)') Comment(10)
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_write
      READ(N_input,'(A)') Buffer
      READ(N_input,*) N_dam_incr
C
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_effstress
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_EVF
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_L
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_Restart
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) IFlag_Interface
      IF (IFlag_Interface.eq.1) Then
        Number_of_Constituents = 3
      ELSE
        Number_of_Constituents = 2
      ENDIF
      READ(N_input,'(A)') Buffer
C -----------------------------------------------------------------------
C
      IF (IFlag_Restart.ne.0.and.IFlag_Restart.ne.1) THEN
        WRITE(N_output,'(A/A,I,A)') 'INPUT format error !!!',
     +       'Your Flag for Restart is ',IFlag_Restart,
     +       ' instead of 0 or 1!!'
        STOP
      ENDIF
C
      If (IFlag_Restart.eq.0) THEN
        OPEN(Phi_dat,FILE=FILE_2,STATUS='UNKNOWN')
        OPEN(Check_c(1),FILE=File_m,STATUS='UNKNOWN')
        OPEN(Check_c(2),FILE=File_f,STATUS='UNKNOWN')
        OPEN(Check_c(3),FILE=File_i,STATUS='UNKNOWN')
        OPEN(SS_dat,FILE='stress_strain.dat',STATUS='UNKNOWN')
      ELSE
        OPEN(Phi_dat,FILE=FILE_2,ACCESS='APPEND',STATUS='OLD')
        OPEN(Check_c(1),FILE=FILE_m,ACCESS='APPEND',STATUS='OLD')
        OPEN(Check_c(2),FILE=FILE_f,ACCESS='APPEND',STATUS='OLD')
        OPEN(Check_c(3),FILE=FILE_i,ACCESS='APPEND',STATUS='OLD')
        OPEN(SS_dat,FILE='stress_strain.dat',ACCESS='APPEND',
     +                                                STATUS='OLD')
      ENDIF
C -----------------------------------------------------------------------
      READ(N_input,*) Number_of_Cycles
      READ(N_input,'(A)') Buffer
      READ(N_input,*) (Sigma_Mean(1,i),i=1,3)
      READ(N_input,*) (Sigma_Mean(2,i),i=1,3)
```

```
      READ(N_input,*) (Sigma_Mean(3,i),i=1,3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) (Sigma_Amp(1,i),i=1,3)
      READ(N_input,*) (Sigma_Amp(2,i),i=1,3)
      READ(N_input,*) (Sigma_Amp(3,i),i=1,3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Number_of_discretization_points
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Damage_stepsize_mean_stress
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Damage_stepsize_cyclic
C -----------------------------------------------------------------------
      write(N_output,'(10(A/)/)') 'Equation of Xi_m',
     +                            Comment(1)(1:LEN(Comment(1))),
     +                            Comment(2)(1:LEN(Comment(2))),
     +                            Comment(3)(1:LEN(Comment(3))),
     +                            Comment(4)(1:LEN(Comment(4))),
     +                            Comment(5)(1:LEN(Comment(5))),
     +                            'Equation of Xi_f',
     +                            Comment(6)(1:LEN(Comment(6))),
     +                            Comment(7)(1:LEN(Comment(7))),
     +                            Comment(8)(1:LEN(Comment(8))),
     +                            Comment(9)(1:LEN(Comment(9))),
     +                            Comment(10)(1:LEN(Comment(10)))
C
      WRITE(N_output,'(A/A/A/)')
     + '-----------------------------------------------------------'
     + ,'Input Data','----------'
      IF (IFlag_EVF.eq.0) THEN
        WRITE(N_output,'(A/)') '---> Initial volume fractions used !!'
      ELSEIF (IFlag_EVF.eq.1) THEN
        WRITE(N_output,'(A/)') '---> Effective volume fractions used !!'
      ENDIF
      IF (IFlag_effstress.eq.0) THEN
        WRITE(N_output,'(A/)')
     +       '---> Concentration tensors not adjusted !!'
      ELSEIF (IFlag_effstress.eq.1) THEN
        WRITE(N_output,'(A/)')
     +       '---> Effective concentration tensors used !!'
      ENDIF
      IF (IFlag_Restart.eq.0) THEN
        WRITE(N_output,'(A/A)') 'Loading Parameters',
     +                          '-----------------'
      ELSEIF (IFlag_Restart.eq.1) THEN
        WRITE(N_output,'(A/A)') 'Loading Parameters --- Restart',
     +                          '-----------------------------'
      ENDIF
      WRITE(N_output,'(3x,A//3(3(F12.4,3x)/))')
     +      'Mean Stress (MPa):',
     +      ((Sigma_Mean(i,j),j=1,3),i=1,3)
      WRITE(N_output,'(3x/3x,A//3(3(F12.4,3x)/))')
     +      'Stress Amplitude (MPa):',
     +      ((Sigma_Amp(i,j),j=1,3),i=1,3)
      WRITE(N_output,'(3x,A,I12)')
     +      'Number of Discretization Points (per Curve):           ',
     +      Number_of_discretization_points
      WRITE(N_output,'(3x,A,d12.6)')
     +      'Stress increment in case of damage (mean stress, MPa):  ',
     +      Damage_stepsize_mean_stress
      WRITE(N_output,'(3x,A,d12.6//)')
     +      'Increment in case of damage (cyclic range, in degrees): ',
     +      Damage_stepsize_cyclic
C -----------------------------------------------------------------------
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Mat_id
C -----------------------------------------------------------------------
      IF (Mat_id.eq.1) WRITE(N_output,'(A/A)')
     +                               'Isotropic Material Data',
     +                               '-----------------------'
      IF (Mat_id.eq.2) WRITE(N_output,'(A/A)')
     +                         'Transversely Isotropic Material Data',
     +                         '-----------------------------------'
      IF (Mat_id.eq.3) WRITE(N_output,'(A/A)')
     +                               'Anisotropic Material Data',
     +                               '------------------------'
C -----------------------------------------------------------------------
C
C     --- Fiber properties ---
C
      READ(N_input,'(A)') Buffer
      READ(N_input,*) c_f_0
      c_m_0 = 1.d0 - c_f_0
      READ(N_input,*) Prop(1)
      READ(N_input,*) Prop(2)
```

```
      READ(N_input,*) Prop(3)
      READ(N_input,*) Prop(4)
      READ(N_input,*) Prop(5)
      READ(N_input,*) Prop(6)
      READ(N_input,*) Prop(7)
      READ(N_input,*) Prop(8)
      READ(N_input,*) Prop(9)
      CALL DTRANS_A_B_1(Prop,Prop_f,9)
C
      IF (IFlag_Restart.eq.0) THEN
C
        CALL MATERIAL(Mat_id,Prop,E_f_bar,Nu_f)
C
        CALL DA_ijkl_LINV(E_f_bar,E_f_bar_inv)
C
      ENDIF
C
      WRITE(N_output,'(/2x,A)') 'Fiber:'
      WRITE(N_output,'(3x,A,F13.3)') 'c_f_0 = ',c_f_0
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_12 = ',Prop(1)
      WRITE(N_output,'(3x,A,F13.3)') 'E_1   = ',Prop(2)
      WRITE(N_output,'(3x,A,F13.3)') 'G_12  = ',Prop(3)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_13 = ',Prop(4)
      WRITE(N_output,'(3x,A,F13.3)') 'E_2   = ',Prop(5)
      WRITE(N_output,'(3x,A,F13.3)') 'G_13  = ',Prop(6)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_23 = ',Prop(7)
      WRITE(N_output,'(3x,A,F13.3)') 'E_3   = ',Prop(8)
      WRITE(N_output,'(3x,A,F13.3)') 'G_23  = ',Prop(9)
C
C     --- Matrix properties ---
C
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Prop(1)
      READ(N_input,*) Prop(2)
      READ(N_input,*) Prop(3)
      READ(N_input,*) Prop(4)
      READ(N_input,*) Prop(5)
      READ(N_input,*) Prop(6)
      READ(N_input,*) Prop(7)
      READ(N_input,*) Prop(8)
      READ(N_input,*) Prop(9)
      CALL DTRANS_A_B_1(Prop,Prop_m,9)
C
      IF (IFlag_Restart.eq.0) THEN
C
        CALL MATERIAL(Mat_id,Prop,E_m_bar,Nu_m)
C
C         - Calculate E_bar_inv
C
        CALL DA_ijkl_LINV(E_m_bar,E_m_bar_inv)
C
      ENDIF
C
      WRITE(N_output,'(/2x,A)') 'Matrix:'
      WRITE(N_output,'(3x,A,F13.3)') 'c_m   = ',c_m_0
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_12 = ',Prop(1)
      WRITE(N_output,'(3x,A,F13.3)') 'E_1   = ',Prop(2)
      WRITE(N_output,'(3x,A,F13.3)') 'G_12  = ',Prop(3)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_13 = ',Prop(4)
      WRITE(N_output,'(3x,A,F13.3)') 'E_2   = ',Prop(5)
      WRITE(N_output,'(3x,A,F13.3)') 'G_13  = ',Prop(6)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_23 = ',Prop(7)
      WRITE(N_output,'(3x,A,F13.3)') 'E_3   = ',Prop(8)
      WRITE(N_output,'(3x,A,F13.3)') 'G_23  = ',Prop(9)
C
C     --- Interface properties ---
C
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Prop(1)
      READ(N_input,*) Prop(2)
      READ(N_input,*) Prop(3)
      READ(N_input,*) Prop(4)
      READ(N_input,*) Prop(5)
      READ(N_input,*) Prop(6)
      READ(N_input,*) Prop(7)
      READ(N_input,*) Prop(8)
      READ(N_input,*) Prop(9)
      CALL DTRANS_A_B_1(Prop,Prop_i,9)
C
      IF (IFlag_Restart.eq.0) THEN
C
        CALL MATERIAL(Mat_id,Prop,E_i_bar,Nu_i)
C
C         - Calculate E_bar_inv
```

```
C
        CALL DA_ijkl_LINV(E_i_bar,E_i_bar_inv)
C
      ENDIF
C
      WRITE(N_output,'(/2x,A)') 'Interface:'
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_12 = ',Prop(1)
      WRITE(N_output,'(3x,A,F13.3)') 'E_1   = ',Prop(2)
      WRITE(N_output,'(3x,A,F13.3)') 'G_12  = ',Prop(3)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_13 = ',Prop(4)
      WRITE(N_output,'(3x,A,F13.3)') 'E_2   = ',Prop(5)
      WRITE(N_output,'(3x,A,F13.3)') 'G_13  = ',Prop(6)
      WRITE(N_output,'(3x,A,F13.3)') 'Nu_23 = ',Prop(7)
      WRITE(N_output,'(3x,A,F13.3)') 'E_3   = ',Prop(8)
      WRITE(N_output,'(3x,A,F13.3)') 'G_23  = ',Prop(9)
C
C     Read damage parameters for the fiber and the matrix
C
        WRITE(N_output,'(/A/A/2x,A)') 'Damage Parameters',
     +                          '----------------',
     +                          'Matrix:'
      READ(N_input,'(A)') Buffer
      READ(N_input,'(A)') Buffer
      READ(N_input,*) F_power_m
        WRITE(N_output,'(/4x,A,d12.4)') 'Power of F  =',F_power_m
      READ(N_input,'(A)') Buffer
      READ(N_input,*) c_Gamma_m
        WRITE(N_output,'(4x,A,d12.4)') 'c_Gamma_m   =',c_Gamma_m
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Kappa_m
        WRITE(N_output,'(4x,A,d12.4)') 'Kappa_m_0   =',Kappa_m
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Lambda_m(1),Lambda_m(2),Lambda_m(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(1,1) =',Lambda_m(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(2,2) =',Lambda_m(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(3,3) =',Lambda_m(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Eta_m(1),Eta_m(2),Eta_m(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(1,1)    =',Eta_m(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(2,2)    =',Eta_m(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(3,3)    =',Eta_m(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Xi_m_0(1), Xi_m_0(2), Xi_m_0(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(1,1)     =',Xi_m_0(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(2,2)     =',Xi_m_0(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(3,3)     =',Xi_m_0(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) dXi_m1(1), dXi_m1(2), dXi_m1(3)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(1,1)  =',dXi_m1(1)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(2,2)  =',dXi_m1(2)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(3,3)  =',dXi_m1(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) dXi_m2(1), dXi_m2(2), dXi_m2(3)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(1,1)  =',dXi_m2(1)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(2,2)  =',dXi_m2(2)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(3,3)  =',dXi_m2(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) a_Matrix
        WRITE(N_output,'(4x,A,d12.4)') 'a_m         =',a_Matrix
      READ(N_input,'(A)') Buffer
      READ(N_input,*) (Vi_m(i),i=1,3)
        WRITE(N_output,'(4x,A,d12.4)') 'Vi(1,1)     =',Vi_m(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Vi(2,2)     =',Vi_m(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Vi(3,3)     =',Vi_m(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) (Phi_m_cr(1,j),j=1,3)
      READ(N_input,*) (Phi_m_cr(2,j),j=1,3)
      READ(N_input,*) (Phi_m_cr(3,j),j=1,3)
        WRITE(N_output,'(/3x,A/3(/3(F10.4,3x)))')
     +    '- Critical Damage Parameters:',((Phi_m_cr(i,j),j=1,3),i=1,3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) N_m1
      READ(N_input,*) N_m2
        WRITE(N_output,'(/3x,A/2(/4x,A,I6))')
     +      '- Limit Values for Cycles',
     +      'Lower Limit = ',N_m1,'Upper Limit = ',N_m2
C
      READ(N_input,'(A)') Buffer
        WRITE(N_output,'(/2x,A)') 'Fiber:'
      READ(N_input,'(A)') Buffer
      READ(N_input,*) F_power_f
        WRITE(N_output,'(/4x,A,d12.4)') 'Power of F  =',F_power_f
      READ(N_input,'(A)') Buffer
      READ(N_input,*) c_Gamma_f
```

```
      WRITE(N_output,'(4x,A,d12.4)') 'c_Gamma_f   =',c_Gamma_f
    READ(N_input,'(A)') Buffer
    READ(N_input,*) Kappa_f
      WRITE(N_output,'(4x,A,d12.4)') 'Kappa_f_0   =',Kappa_f
    READ(N_input,'(A)') Buffer
    READ(N_input,*) Lambda_f(1),Lambda_f(2),Lambda_f(3)
      WRITE(N_output,'(4x,A,d12.4)') 'Lambda(1,1) =',Lambda_f(1)
      WRITE(N_output,'(4x,A,d12.4)') 'Lambda(2,2) =',Lambda_f(2)
      WRITE(N_output,'(4x,A,d12.4)') 'Lambda(3,3) =',Lambda_f(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) Eta_f(1),Eta_f(2),Eta_f(3)
      WRITE(N_output,'(4x,A,d12.4)') 'Eta(1,1)    =',Eta_f(1)
      WRITE(N_output,'(4x,A,d12.4)') 'Eta(2,2)    =',Eta_f(2)
      WRITE(N_output,'(4x,A,d12.4)') 'Eta(3,3)    =',Eta_f(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) Xi_f_0(1),Xi_f_0(2),Xi_f_0(3)
      WRITE(N_output,'(4x,A,d12.4)') 'Xi(1,1)     =',Xi_f_0(1)
      WRITE(N_output,'(4x,A,d12.4)') 'Xi(2,2)     =',Xi_f_0(2)
      WRITE(N_output,'(4x,A,d12.4)') 'Xi(3,3)     =',Xi_f_0(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) dXi_f1(1),dXi_f1(2),dXi_f1(3)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(1,1)  =',dXi_f1(1)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(2,2)  =',dXi_f1(2)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(3,3)  =',dXi_f1(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) dXi_f2(1), dXi_f2(2), dXi_f2(3)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(1,1)  =',dXi_f2(1)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(2,2)  =',dXi_f2(2)
      WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(3,3)  =',dXi_f2(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) a_Fiber
      WRITE(N_output,'(4x,A,d12.4)') 'a_f         =',a_Fiber
    READ(N_input,'(A)') Buffer
    READ(N_input,*) (Vi_f(i),i=1,3)
      WRITE(N_output,'(4x,A,d12.4)') 'Vi(1,1)     =',Vi_f(1)
      WRITE(N_output,'(4x,A,d12.4)') 'Vi(2,2)     =',Vi_f(2)
      WRITE(N_output,'(4x,A,d12.4)') 'Vi(3,3)     =',Vi_f(3)
    READ(N_input,'(A)') Buffer
    READ(N_input,*) (Phi_f_cr(1,j),j=1,3)
    READ(N_input,*) (Phi_f_cr(2,j),j=1,3)
    READ(N_input,*) (Phi_f_cr(3,j),j=1,3)
      WRITE(N_output,'(/3x,A/3(/3(F10.4,3x)))')
     +    '- Critical Damage Parameters:',((Phi_f_cr(i,j),j=1,3),i=1,3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) N_f1
      READ(N_input,*) N_f2
        WRITE(N_output,'(/3x,A/2(/4x,A,I6))')
     +      '- Limit Values for Cycles',
     +      'Lower Limit = ',N_f1,'Upper Limit = ',N_f2
C
      READ(N_input,'(A)') Buffer
        WRITE(N_output,'(/2x,A)') 'Interface:'
      READ(N_input,'(A)') Buffer
      READ(N_input,*) F_power_i
        WRITE(N_output,'(/4x,A,d12.4)') 'Power of F  =',F_power_i
      READ(N_input,'(A)') Buffer
      READ(N_input,*) c_Gamma_i
        WRITE(N_output,'(4x,A,d12.4)') 'c_Gamma_i   =',c_Gamma_i
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Kappa_i
        WRITE(N_output,'(4x,A,d12.4)') 'Kappa_i_0   =',Kappa_i
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Lambda_i(1),Lambda_i(2),Lambda_i(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(1,1) =',Lambda_i(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(2,2) =',Lambda_f(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Lambda(3,3) =',Lambda_f(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Eta_i(1),Eta_i(2),Eta_i(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(1,1)    =',Eta_i(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(2,2)    =',Eta_i(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Eta(3,3)    =',Eta_i(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) Xi_i_0(1),Xi_i_0(2),Xi_i_0(3)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(1,1)     =',Xi_i_0(1)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(2,2)     =',Xi_i_0(2)
        WRITE(N_output,'(4x,A,d12.4)') 'Xi(3,3)     =',Xi_i_0(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) dXi_i1(1),dXi_i1(2),dXi_i1(3)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(1,1)  =',dXi_i1(1)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(2,2)  =',dXi_i1(2)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_1(3,3)  =',dXi_i1(3)
      READ(N_input,'(A)') Buffer
      READ(N_input,*) dXi_i2(1), dXi_i2(2), dXi_i2(3)
        WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(1,1)  =',dXi_i2(1)
```

```
667         WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(2,2)  =',dXi_i2(2)
668         WRITE(N_output,'(4x,A,d12.4)') 'dXi_2(3,3)  =',dXi_i2(3)
669       READ(N_input,'(A)') Buffer
670       READ(N_input,*) a_Interface
671         WRITE(N_output,'(4x,A,d12.4)') 'a_i         =',a_Interface
672       READ(N_input,'(A)') Buffer
673       READ(N_input,*) (Vi_i(i),i=1,3)
674         WRITE(N_output,'(4x,A,d12.4)') 'Vi(1,1)     =',Vi_i(1)
675         WRITE(N_output,'(4x,A,d12.4)') 'Vi(2,2)     =',Vi_i(2)
676         WRITE(N_output,'(4x,A,d12.4)') 'Vi(3,3)     =',Vi_i(3)
677       READ(N_input,'(A)') Buffer
678       READ(N_input,*) (Phi_i_cr(1,j),j=1,3)
679       READ(N_input,*) (Phi_i_cr(2,j),j=1,3)
680       READ(N_input,*) (Phi_i_cr(3,j),j=1,3)
681         WRITE(N_output,'(/3x,A/3(/3(F10.4,3x)))')
682      +    '- Critical Damage Parameters:',((Phi_i_cr(i,j),j=1,3),i=1,3)
683       READ(N_input,'(A)') Buffer
684       READ(N_input,*) N_i1
685       READ(N_input,*) N_i2
686         WRITE(N_output,'(/3x,A/2(/4x,A,I6))')
687      +      '- Limit Values for Cycles',
688      +      'Lower Limit = ',N_i1,'Upper Limit = ',N_i2
689  C
690       CLOSE(N_output)
691  C
692  C ---------------------------------------------------------------------
693  C ---------------------------------------------------------------------
694  C                          R E S T A R T
695  C ---------------------------------------------------------------------
696  C ---------------------------------------------------------------------
697       N_inc_old = 0
698  C
699       IF (IFlag_Restart.eq.1) Then
700  C
701  C ----- Read number of cycles for previous loading
702  C
703         READ(N_Restart,*) N_data_group
704         READ(N_Restart,*) Number_of_Restarts
705         READ(N_Restart,*) Number_of_Cycles_old
706         READ(N_Restart,*) N_inc_old
707  C
708  C ----- Read previous mean stress
709  C
710         READ(N_Restart,*) N_data_group
711         READ(N_Restart,*) dSigma_tot_old
712         READ(N_Restart,*) (Sigma_mean_old(1,j),j=1,3)
713         READ(N_Restart,*) (Sigma_mean_old(2,j),j=1,3)
714         READ(N_Restart,*) (Sigma_mean_old(3,j),j=1,3)
715  C
716  C ----- Read original material properties which did not change
717  C
718         READ(N_Restart,*) N_data_group
719         READ(N_Restart,*) c_f,c_m
720         READ(N_Restart,*) N_data_group
721         READ(N_Restart,*) (Prop(i),i=1,9)
722         READ(N_Restart,*) ((Nu_m(i,j),j=1,3),i=1,3)
723         READ(N_Restart,*) N_data_group
724         READ(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
725      +                  ((((E_m_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
726         READ(N_Restart,*) N_data_group
727         READ(N_Restart,*) (Prop(i),i=1,9)
728         READ(N_Restart,*) ((Nu_f(i,j),j=1,3),i=1,3)
729         READ(N_Restart,*) N_data_group
730         READ(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
731      +                  ((((E_f_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
732         READ(N_Restart,*) N_data_group
733         READ(N_Restart,*) (Prop(i),i=1,9)
734         READ(N_Restart,*) ((Nu_i(i,j),j=1,3),i=1,3)
735         READ(N_Restart,*) N_data_group
736         READ(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
737      +                  ((((E_i_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
738  C
739  C ----- Read previous damage parameters - Matrix - unaltered
740  C
741         READ(N_Restart,*) N_data_group
742         READ(N_Restart,*) F_power_m
743         READ(N_Restart,*) c_Gamma_m
744         READ(N_Restart,*) (Lambda_m(j),j=1,3)
745         READ(N_Restart,*) (Eta_m(j),j=1,3)
746         READ(N_Restart,*) (Vi_m(j),j=1,3)
747  C
748  C ----- Read damage parameters which have been updated - matrix
749  C
750         READ(N_Restart,*) N_data_group
```

```
      READ(N_Restart,*) (Xi_m(j),j=1,3)
      READ(N_Restart,*) Kappa_m
      READ(N_Restart,*) (Phi_m(1,j),j=1,3)
      READ(N_Restart,*) (Phi_m(2,j),j=1,3)
      READ(N_Restart,*) (Phi_m(3,j),j=1,3)
      READ(N_Restart,*) g_m
      READ(N_Restart,*) ((Gamma_m(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) ((Y_m(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) F_pot_m
C
C ----- Read previous damage parameters - Fiber - unaltered
C
      READ(N_Restart,*) N_data_group
      READ(N_Restart,*) F_power_f
      READ(N_Restart,*) c_Gamma_f
      READ(N_Restart,*) (Lambda_f(j),j=1,3)
      READ(N_Restart,*) (Eta_f(j),j=1,3)
      READ(N_Restart,*) (Vi_f(j),j=1,3)
C
C ----- Read damage parameters which have been updated - fiber
C
      READ(N_Restart,*) N_data_group
      READ(N_Restart,*) (Xi_f(j),j=1,3)
      READ(N_Restart,*) Kappa_f
      READ(N_Restart,*) (Phi_f(1,j),j=1,3)
      READ(N_Restart,*) (Phi_f(2,j),j=1,3)
      READ(N_Restart,*) (Phi_f(3,j),j=1,3)
      READ(N_Restart,*) g_f
      READ(N_Restart,*) ((Gamma_f(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) ((Y_f(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) F_pot_f
C
C ----- Read previous damage parameters - Interface - unaltered
C
      READ(N_Restart,*) N_data_group
      READ(N_Restart,*) F_power_i
      READ(N_Restart,*) c_Gamma_i
      READ(N_Restart,*) (Lambda_i(j),j=1,3)
      READ(N_Restart,*) (Eta_i(j),j=1,3)
      READ(N_Restart,*) (Vi_i(j),j=1,3)
C
C ----- Read damage parameters which have been updated - fiber
C
      READ(N_Restart,*) N_data_group
      READ(N_Restart,*) (Xi_i(j),j=1,3)
      READ(N_Restart,*) Kappa_i
      READ(N_Restart,*) (Phi_i(1,j),j=1,3)
      READ(N_Restart,*) (Phi_i(2,j),j=1,3)
      READ(N_Restart,*) (Phi_i(3,j),j=1,3)
      READ(N_Restart,*) g_i
      READ(N_Restart,*) ((Gamma_i(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) ((Y_i(i,j),j=1,3),i=1,3)
      READ(N_Restart,*) F_pot_i
      REWIND(N_Restart)
C ---------------------------------------------------------------------
      CALL DA_ijkl_LINV(E_m_bar,E_m_bar_inv)
C ---------------------------------------------------------------------
      CALL DA_ijkl_LINV(E_f_bar,E_f_bar_inv)
      CALL DA_ijkl_LINV(E_i_bar,E_i_bar_inv)
C
    ENDIF
C
C ---------------------------------------------------------------------
C   Calculation of Lambda as Lame's constant:
C ---------------------------------------------------------------------
C
    If (Mat_id.eq.1) THEN
C
      Lambda_f(1) = Prop_f(1) * Prop_f(2) /
     +          ( ( 1.d0 + Prop_f(1) ) * ( 1.d0 - 2.d0 * Prop_f(1) ) )
      Lambda_f(2) = Lambda_f(1)
      Lambda_f(3) = Lambda_f(1)
      Lambda_m(1) = Prop_m(1) * Prop_m(2) /
     +          ( ( 1.d0 + Prop_m(1) ) * ( 1.d0 - 2.d0 * Prop_m(1) ) )
      Lambda_m(2) = Lambda_m(1)
      Lambda_m(3) = Lambda_m(1)
C
      IF (IFlag_Interface.eq.1) THEN
C
        Lambda_i(1) = Prop_i(1) * Prop_i(2) /
     +          ( ( 1.d0 + Prop_i(1) ) * ( 1.d0 - 2.d0 * Prop_i(1) ) )
        Lambda_i(2) = Lambda_i(1)
        Lambda_i(3) = Lambda_i(1)
C
```

```
        ENDIF
C
      ENDIF
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C     Print Headers for checking
C
      IF (IFlag_restart.eq.0) THEN
C
        WRITE(Check_c(1),'(7(A,f15.5,3x/))') 'c_m    =',c_m_0,
     +                                  'Lamdba =',Lambda_m(1),
     +                                  'Eta    =',Eta_m(1),
     +                                  'Xi     =',Xi_m_0(1),
     +                                  'dXi_1  =',dXi_m1(1),
     +                                  'dXi_2  =',dXi_m2(1),
     +                                  'v      =',Vi_m(1)
        WRITE(Check_c(1),'(6x,A,11x,A,14x,A,15x,A,13x,A,11x,A,9x,
     +                                  A,9x,A,6x,A,8x,A,7x,A)')
     +       'ITER','g','Y','F','Phi_11',
     +       'Xi','Sigma_11','Kappa','c_m','w','Sigma_tot(1,1)'
        WRITE(Check_c(2),'(7(A,f15.5,3x/))') 'c_f    = ',c_f_0,
     +                                  'Lamdba = ',Lambda_f(1),
     +                                  'Eta    = ',Eta_f(1),
     +                                  'Xi     = ',Xi_f_0(1),
     +                                  'dXi_1  = ',dXi_f1(1),
     +                                  'dXi_2  = ',dXi_f2(1),
     +                                  'v      = ',Vi_f(1)
        WRITE(Check_c(2),'(6x,A,11x,A,14x,A,15x,A,13x,A,11x,A,9x,
     +                                          A,9x,A,6x,A,8x,A)')
     +         'ITER','g','Y','F','Phi_11',
     +         'Xi','Sigma_11','Kappa','c_f','w'
C
        If (IFlag_Interface.eq.1)
     +    WRITE(Check_c(3),'(2x,6(A,f15.5,3x/),
     +       6x,A,11x,A,14x,A,15x,A,13x,A,11x,A,9x,A,9x,A,8x,A)')
     +                                  'Lamdba = ',Lambda_i(1),
     +                                  'Eta    = ',Eta_i(1),
     +                                  'Xi     = ',Xi_i_0(1),
     +                                  'dXi_1  = ',dXi_i1(1),
     +                                  'dXi_2  = ',dXi_i2(1),
     +                                  'v      = ',Vi_i(1),
     +         'ITER','g','Y','F','Phi_11','Xi','Sigma_11','Kappa','w'
C
      ENDIF
C -----------------------------------------------------------------------
C     Header for damage output
C -----------------------------------------------------------------------
      IF (IFlag_Restart.eq.0) THEN
        IF (IFlag_Interface.eq.1) THEN
          WRITE(Phi_dat,'(2x,A,3x,A,5x,A,2x,A,6x,A,5x,A,2x,A,6x,A,5x,A,
     +                  3x,A,6x,A,6x,A)')
     +    'Cycle','Phi_m(1,1)','Sigma_m(1,1)',' Xi_m(1,1)','Phi_f(1,1)',
     +    'Sigma_f(1,1)',' Xi_f(1,1)',
     +    'Phi_i(1,1)','Sigma_i(1,1)','Xi_i(1,1)',
     +    'Phi_c(1,1)','Sigma_tot(1,1)'
          write(Phi_dat,'(1x,I6,11(1x,d14.8))')
     +         0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
        ELSEIF (IFlag_Interface.eq.0) THEN
          WRITE(Phi_dat,'(2x,A,5x,A,7x,A,4x,A,8x,A,7x,A,4x,A,
     +                                          8x,A,5x,A)')
     +    'Cycle','Phi_m(1,1)','Sigma_m(1,1)',' Xi_m(1,1)','Phi_f(1,1)',
     +    'Sigma_f(1,1)',' Xi_f(1,1)','Phi_c(1,1)','Sigma_tot(1,1)'
          write(Phi_dat,'(1x,I6,8(3x,d14.8))')
     +         0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
        ENDIF
      ENDIF
C
C -----------------------------------------------------------------------
C
C         S T A R T   o f   L o a d i n g   t o
C
C                     M e a n   S t r e s s   L e v e l
C
C -----------------------------------------------------------------------
C
C     Initialize variables:
C
      IF (IFlag_restart.eq.1) THEN
C
        IF (Sigma_mean_old(1,1).lt.Sigma_mean(1,1)) THEN
C
          CALL DTRANS_A_B_2(Sigma_mean,dSigma_tot_0,3)
          dum = 1.d0 / Number_of_discretization_points
```

```
          CALL Dc_A_ij(dum,dSigma_tot_0,3)
          CALL DTRANS_A_B_2(Sigma_mean_old,Sigma_tot,3)
C
        ENDIF
C
C ----- Set the damage increment equal to 0 upon restart
C       All other necessary variables have been read and hence have
C       the appropriate values
C
        CALL DINITIALIZE_ZERO_2(dPhi_m,3)
        CALL DINITIALIZE_ZERO_2(dPhi_f,3)
        CALL DINITIALIZE_ZERO_2(dPhi_i,3)
        NOC_old = Number_Of_Cycles_old
C
      ELSE
C
C ----- Initialization of variables upon new loading
C
        dum = 1.d0 / Number_of_discretization_points
        CALL DTRANS_A_B_2(Sigma_mean,dSigma_tot_0,3)
        CALL Dc_A_ij(dum,dSigma_tot_0,3)
C
        Number_of_Restarts = 0
C
        CALL DINITIALIZE_ZERO_2(Sigma_tot,3)
        CALL DINITIALIZE_ZERO_2(Y_m,3)
        CALL DINITIALIZE_ZERO_2(Y_f,3)
        CALL DINITIALIZE_ZERO_2(Y_i,3)
        g_m                 = -1.d0
        g_f                 = -1.d0
        g_i                 = -1.d0
        CALL DINITIALIZE_ZERO_2(Phi_m,3)
        CALL DINITIALIZE_ZERO_2(Phi_f,3)
        CALL DINITIALIZE_ZERO_2(Phi_i,3)
        CALL DINITIALIZE_ZERO_2(dPhi_m,3)
        CALL DINITIALIZE_ZERO_2(dPhi_f,3)
        CALL DINITIALIZE_ZERO_2(dPhi_i,3)
        alpha   = 0.d0
        F_pot_m = 0.d0
        F_pot_f = 0.d0
        F_pot_i = 0.d0
        CALL DINITIALIZE_ZERO_2(Gamma_m,3)
        CALL DINITIALIZE_ZERO_2(Gamma_f,3)
        CALL DINITIALIZE_ZERO_2(Gamma_i,3)
        Incr   = 0
        Icount = 0
        NOC_old = 0
C
      ENDIF
C
C ---------------------------------------------------------------------
C     Set the stepsize in the damage region in dSigma_damage. The coefficient
C     Damage_stepsize controls the stepsize since the loop only determines the
C     maximum component of the cyclic amplitude and then divides the stress
C     amplitude by the absolute maximum value.
C     I_max, J_max    ===> locate the position of the maximum stress component
C
      Sigma_max = 0.d0
C
      DO i=1, 3
        DO j=1, 3
          IF (DABS(Sigma_mean(i,j)).gt.Sigma_max) THEN
            Sigma_max = DABS(Sigma_mean(i,j))
            I_max = i
            J_max = j
          ENDIF
        ENDDO
      ENDDO
      DO i=1, 3
        DO j=1, 3
          dSigma_damage(i,j) = Damage_Stepsize *
     +                                 Sigma_mean(i,j)/Sigma_max
        ENDDO
      ENDDO
C
      IFlag_fail = 0
C ---------------------------------------------------------------------
      IF (IFlag_restart.eq.0) THEN
        L_inc = 0
      ELSEIF (IFlag_restart.eq.1) THEN
        L_inc = N_inc_old
      ENDIF
C \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
C
```

```
C     --- Now start loading increment loop
C
C \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
C
      IF (IFlag_Restart.eq.0.or.
     +      (IFlag_Restart.eq.1.and.
     +                 Sigma_mean(1,1).gt.Sigma_mean_old(1,1))) THEN
C
C --- dSigma_tot > 0 indicates restart with the old mean stress being
C     lower then the new mean stress
C
        If (IFlag_Restart.eq.1)
     +     write(check_r,'(A)') 'Restart with increase in mean stress'
C ----------------------------------------------------------------------
C
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
C
        DO 1000 WHILE(DABS(Sigma_mean(1,1)-Sigma_tot(1,1)).gt.1.d-5)
C
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
          L_inc          = L_inc +1
          STEP_factor    = 1.d0
          IFLag_min_step = 0
C ----------------------------------------------------------------------
          IF (IFlag_Damage.gt.0) THEN
C ----------------------------------------------------------------------
C ----------------------------------------------------------------------
C           > 0 ---> damage occurred in previous increment
C
C           Check if the difference between the current and the mean stress
C           is larger than the fixed value for the damage region
C ----------------------------------------------------------------------
            IF (DABS(Sigma_mean(1,1)-Sigma_tot(1,1)).gt.
     +                                  dSigma_damage(1,1)) THEN
C ----------------------------------------------------------------------
C           --- Set the stress increment to the fixed value for the damage
C               region.
C ----------------------------------------------------------------------
              CALL DTRANS_A_B_2(dSigma_damage,dSigma_tot,3)
              IFlag_min_step = 1
C ----------------------------------------------------------------------
            ELSE
C ----------------------------------------------------------------------
C           --- Set the stress increment to the difference between the
C               two stresses
C ----------------------------------------------------------------------
              CALL DA_ij_M_B_ij(Sigma_mean,Sigma_tot,dSigma_tot,3)
              IFlag_min_step = 1
C ----------------------------------------------------------------------
            ENDIF
C ----------------------------------------------------------------------
C ----------------------------------------------------------------------
          ELSEIF (IFlag_Damage.eq.0) THEN
C ----------------------------------------------------------------------
C           0 ---> indicates no damage
C
C           Check if the difference between the current and the mean stress
C           is larger than the fixed value for the damage region or the
C           standard stepsize
C ----------------------------------------------------------------------
            IF (DABS(Sigma_mean(1,1)-Sigma_tot(1,1)).gt.
     +                                  dSigma_damage(1,1)) THEN
C ----------------------------------------------------------------------
              IF (DABS(Sigma_mean(1,1)-Sigma_tot(1,1)).gt.
     +                                  dSigma_tot_0(1,1)) THEN
C ----------------------------------------------------------------------
C           --- Set the stress increment to the standard stepsize
C ----------------------------------------------------------------------
                CALL DTRANS_A_B_2(dSigma_tot_0,dSigma_tot,3)
C ----------------------------------------------------------------------
              ELSE
C ----------------------------------------------------------------------
C           --- Set the stress increment to the difference between the two
C ----------------------------------------------------------------------
                CALL DA_ij_M_B_ij(Sigma_mean,Sigma_tot,dSigma_tot,3)
C ----------------------------------------------------------------------
              ENDIF
C ----------------------------------------------------------------------
            ELSE
C ----------------------------------------------------------------------
C           Difference between mean stress and current stress is lower than
C           the fixed stepsize for the damage region
C           ===> Set the stress increment to the difference between the
C                two stresses
```

```
C ------------------------------------------------------------------
            CALL DA_ij_M_B_ij(Sigma_mean,Sigma_tot,dSigma_tot,3)
            IFlag_min_step = 1
C ------------------------------------------------------------------
          ENDIF
C ------------------------------------------------------------------
        ENDIF
C ------------------------------------------------------------------
C
C   ----- Calculate effective volume fractions -------------------------
C
        CALL EFF_VOL_FRAC(c_f,c_m,c_f_0,c_m_0,Phi_f,Phi_m,
     +                                          Phi_f_cr,Phi_m_cr)
C
        c_m_adj = c_m
        c_f_adj = c_f
C
        IF (IFlag_EVF.eq.0) THEN
          c_m     = c_m_0
          c_f     = c_f_0
        ENDIF
C ------------------------------------------------------------------
C       --- Calculate stress and strain concentration tensors:
C ------------------------------------------------------------------
        CALL M_ijkl_4_81(Phi_m,1,0,0,M_m,DM,D2M)
        CALL M_ijkl_4_81(Phi_f,1,0,0,M_f,DM,D2M)
        CALL SSCF(Nu_m,c_f,E_f_bar,E_f_bar_inv,E_m_bar,E_m_bar_inv,S,
     +         M_m,M_f,A_f_bar,B_f_bar,A_m_bar,B_m_bar,A_f,B_f,A_m,B_m)
C ------------------------------------------------------------------
C     Note: The stress has to be amplified by M in order to account for the
C           reduction in effective area/volume
C           ---> calculate M of the composite to adjust the stress before
C                distributing it!!!
C ------------------------------------------------------------------
        CALL M_COMPOSITE(c_f,c_m,M_f,M_m,B_f,B_m,M_comp)
C
C       --- Calculate effective applied stress
C
        CALL DA_ijkl_B_kl(M_comp,Sigma_tot,Sigma_tot_eff,3)
C ------------------------------------------------------------------
        IF (L_inc.eq.1) THEN
C
C       --- Calculate undamaged initial elastic properties of the composite
C
          CALL COMPOSIT_PROP(E_c_0,C_c_0,E_m_bar_inv,E_f_bar_inv,
     +             E_m_bar,E_f_bar,c_f,c_m,A_m,B_m,A_f,B_f,Phi_m,Phi_f)
C
        ENDIF
C
C ------------------------------------------------------------------
C       --- Calculate the stresses in the constituents
C ------------------------------------------------------------------
        IF (IFlag_effstress.eq.1) THEN
C
          CALL DA_ijkl_B_kl(B_m_bar,Sigma_tot_eff,Sigma_m,3)
          CALL DA_ijkl_B_kl(B_f_bar,Sigma_tot_eff,Sigma_f,3)
C
        ELSEIF (IFlag_effstress.eq.0) THEN
C
          CALL DA_ijkl_B_kl(B_m,Sigma_tot_eff,Sigma_m,3)
          CALL DA_ijkl_B_kl(B_f,Sigma_tot_eff,Sigma_f,3)
C
        ENDIF
C
C ------------------------------------------------------------------
C     Calculate the stress acting on the interface as the difference of
C     stress states between the matrix and the fibers
C ------------------------------------------------------------------
          IF (IFlag_Interface.eq.1) THEN
            CALL DA_ij_M_B_ij(Sigma_f,Sigma_m,Sigma_i,3)
          ENDIF
C ------------------------------------------------------------------
C       --- Reset the flags IFlag_min_step, IFlag_step, IFlag_stepsize
C ------------------------------------------------------------------
        CALL I_INITIALIZE_ZERO_1(IFlag_stepsize,3)
        IFlag_step = 0
        N_it = 0
C ------------------------------------------------------------------
C     --- Do loop over the number of constituents to check whether the
C         load increment is acceptable.
C
C ------------------------------------------------------------------
C
        DO 1100 WHILE(IFlag_Step.lt.Number_of_Constituents)
```

```
C
C -----------------------------------------------------------------------
            N_it = N_it + 1
C -----------------------------------------------------------------------
C
            IFlag_step = 0
            CALL Dc_A_ij(STEP_factor,dSigma_tot,3)
C -----------------------------------------------------------------------
C       --- If the stress increment is lower then the fixed damage stress
C           increment but the flag for minimum stepsize has not been set
C           before then set the stress increment to the fixed damage
C           increment and set the flag for minimum stepsize
C -----------------------------------------------------------------------
            IF (dSigma_tot(I_max,J_max).lt.dSigma_damage(I_max,J_max).
     +          and.IFlag_min_step.eq.0) THEN
C
              CALL DTRANS_A_B_2(dSigma_damage,dSigma_tot,3)
              Iflag_min_step = 1
C
            ENDIF
C -----------------------------------------------------------------------
C
C         --- Calculate effective applied stress increment
C
            CALL DA_ijkl_B_kl(M_comp,dSigma_tot,dSigma_tot_eff,3)
C -----------------------------------------------------------------------
C       --- Calculate the stress increments in the constituents
C -----------------------------------------------------------------------
            IF (IFlag_effstress.eq.1) THEN
C
              CALL DA_ijkl_B_kl(B_m_bar,dSigma_tot_eff,dSigma_m,3)
              CALL DA_ijkl_B_kl(B_f_bar,dSigma_tot_eff,dSigma_f,3)
C
            ELSEIF (IFlag_effstress.eq.0) THEN
C
              CALL DA_ijkl_B_kl(B_m,dSigma_tot_eff,dSigma_m,3)
              CALL DA_ijkl_B_kl(B_f,dSigma_tot_eff,dSigma_f,3)
C
            ENDIF
C
            Sigma_m_0 = Sigma_m(1,1) + dSigma_m(1,1)
            Sigma_f_0 = Sigma_f(1,1) + dSigma_f(1,1)
            Sigma_i_0 = Sigma_f_0 - Sigma_m_0
C -----------------------------------------------------------------------
C       --- Calculate the stress increment acting on the interface as the
C           difference of stress states between the matrix and the fibers
C -----------------------------------------------------------------------
            IF (IFlag_Interface.eq.1) THEN
              CALL DA_ij_M_B_ij(dSigma_f,dSigma_m,dSigma_i,3)
            ENDIF
C -----------------------------------------------------------------------
C
            IF (N_misc.gt.0) THEN
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Total stresses:',
     +                        ((Sigma_tot(i,j),j=1,3),i=1,3)
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Constituent stresses: Matrix',
     +                        ((Sigma_m(i,j),j=1,3),i=1,3)
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Constituent stresses: Fibers',
     +                        ((Sigma_f(i,j),j=1,3),i=1,3)
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Total stress increments:',
     +                        ((dSigma_tot(i,j),j=1,3),i=1,3)
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Constituent stress increments: Matrix',
     +                        ((dSigma_m(i,j),j=1,3),i=1,3)
              WRITE(N_misc,'(//A/3(3(3x,d12.4)/))')
     +                    'Constituent stress increments: Fibers',
     +                        ((dSigma_f(i,j),j=1,3),i=1,3)
            ENDIF
C -----------------------------------------------------------------------
C       --- Loop over the number of constituents
C -----------------------------------------------------------------------
C           ID =1  ---> matrix
C           ID =2  ---> fiber
C           ID =3  ---> interface
C -----------------------------------------------------------------------
C
            DO 1110 ID = 1, Number_of_Constituents
C
C -----------------------------------------------------------------------
C           --- Transfer constituent variables to global variables
```

```
C
C ***********************************************************************
            IF (ID.eq.1) THEN
C ***********************************************************************
C             --- Matrix material
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_m_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_m_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_m,Eta,3)
              CALL DTRANS_A_B_1(Lambda_m,Lambda,3)
              do i=1,3
                V(i)  = Vi_m(i)
              enddo
              IF (IFlag_restart.eq.0) THEN
                CALL DTRANS_A_B_1(Xi_m_0,Xi,3)
              ELSEIF (IFlag_restart.eq.1) THEN
                CALL DTRANS_A_B_1(Xi_m,Xi,3)
              ENDIF
              c_Gamma   = c_Gamma_m
              F_power   = F_power_m
C
C             --- Variables which change in the process of computation
C
              CALL DTRANS_A_B_2(Sigma_m,Sigma_old,3)
              CALL DTRANS_A_B_2(dSigma_m,dSigma,3)
              CALL DTRANS_A_B_2(Phi_m,Phi_old,3)
              CALL DTRANS_A_B_2(Gamma_m,Gamma_old,3)
              CALL DTRANS_A_B_2(Y_m,Y_old,3)
              Kappa_old = Kappa_m
              g_old     = g_m
              F_pot_old = F_pot_m
C
C ***********************************************************************
            ELSEIF (ID.eq.2) THEN
C ***********************************************************************
C
C             --- Fiber material ---------------------------------
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_f_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_f_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_f,Eta,3)
              CALL DTRANS_A_B_1(Lambda_f,Lambda,3)
              do i=1,3
                V(i)  = Vi_f(i)
              enddo
              IF (IFlag_restart.eq.0) THEN
                CALL DTRANS_A_B_1(Xi_f_0,Xi,3)
              ELSEIF (IFlag_restart.eq.1) THEN
                CALL DTRANS_A_B_1(Xi_f,Xi,3)
              ENDIF
              c_Gamma   = c_Gamma_f
              F_power   = F_power_f
C
C             --- Variables which change in the process of computation
C
              CALL DTRANS_A_B_2(Sigma_f,Sigma_old,3)
              CALL DTRANS_A_B_2(dSigma_f,dSigma,3)
              CALL DTRANS_A_B_2(Phi_f,Phi_old,3)
              CALL DTRANS_A_B_2(Gamma_f,Gamma_old,3)
              CALL DTRANS_A_B_2(Y_f,Y_old,3)
              Kappa_old = Kappa_f
              g_old     = g_f
              F_pot_old = F_pot_f
C
C ***********************************************************************
            ELSEIF (ID.eq.3) THEN
C ***********************************************************************
C
C             --- Interface --------------------------------------
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_i_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_i_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_i,Eta,3)
              CALL DTRANS_A_B_1(Lambda_i,Lambda,3)
              do i=1,3
                V(i)  = Vi_i(i)
              enddo
              IF (IFlag_restart.eq.0) THEN
```

```
1339                CALL DTRANS_A_B_1(Xi_i_0,Xi,3)
1340              ELSEIF (IFlag_restart.eq.1) THEN
1341                CALL DTRANS_A_B_1(Xi_i,Xi,3)
1342              ENDIF
1343              c_Gamma   = c_Gamma_i
1344              F_power   = F_power_i
1345 C
1346 C             --- Variables which change in the process of computation
1347 C
1348              CALL DTRANS_A_B_2(Sigma_i,Sigma_old,3)
1349              CALL DTRANS_A_B_2(dSigma_i,dSigma,3)
1350              CALL DTRANS_A_B_2(Phi_i,Phi_old,3)
1351              CALL DTRANS_A_B_2(Gamma_i,Gamma_old,3)
1352              CALL DTRANS_A_B_2(Y_i,Y_old,3)
1353              Kappa_old = Kappa_i
1354              g_old     = g_i
1355              F_pot_old = F_pot_i
1356 C
1357 C ***************************************************************
1358            ENDIF
1359 C ***************************************************************
1360            IFLAG_IMP = 0
1361            IMP       = 'NN'
1362 C ---------------------------------------------------------------
1363 C
1364 C         --- Calculate new stress level for each constituent and store
1365 C             dSigma for each iteration in variable dSigma_iter
1366 C ---------------------------------------------------------------
1367            CALL DTRANS_A_B_2(dSigma,dSigma_iter,3)
1368            CALL DA_ij_P_B_ij(Sigma_old,dSigma,Sigma_new,3)
1369 C
1370 C           - Calculate w
1371 C
1372            do i=1, 3
1373              do j=1, 3
1374                w_old(i,j) = Delta(i,j) * ( Lambda(i) * Eta(i) *
1375     +                       DXY((Kappa_old/Lambda(i)),Xi(i)) + V(i) )
1376              enddo
1377            enddo
1378 C
1379            CALL DA_inv_3(w_old,w_inv)
1380 C
1381 C           - Calculate M, dM_dPhi, ddM_dPHI
1382 C
1383            CALL M_ijkl_4_81(Phi_old,1,1,1,M,DM,D2M)
1384 C
1385            CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
1386 C
1387            F_pot_new = F_pot(w_inv,Y_new,Gamma_old)
1388            g_new     = DXY(F_pot_new,F_power) - 1.d0
1389 C ---------------------------------------------------------------
1390 C         --- Store the calculated values for each constituent
1391 C ***************************************************************
1392            IF (ID.eq.1) THEN
1393 C ***************************************************************
1394              CALL DTRANS_A_B_2(w_old,w_old_m,3)
1395              CALL DTRANS_A_B_2(w_inv,w_inv_m,3)
1396              CALL DTRANS_A_B_4(M,M_m,3)
1397              CALL DTRANS_A_B_6(DM,DM_m,3)
1398              CALL DTRANS_A_B_1(D2M,D2M_m,6561)
1399              CALL DTRANS_A_B_2(Y_new,Y_new_m,3)
1400              CALL DTRANS_A_B_2(Sigma_new,Sigma_new_m,3)
1401              F_pot_new_m = F_pot_new
1402              g_new_m     = g_new
1403 C ***************************************************************
1404            ELSEIF (ID.eq.2) THEN
1405 C ***************************************************************
1406              CALL DTRANS_A_B_2(w_old,w_old_f,3)
1407              CALL DTRANS_A_B_2(w_inv,w_inv_f,3)
1408              CALL DTRANS_A_B_4(M,M_f,3)
1409              CALL DTRANS_A_B_6(DM,DM_f,3)
1410              CALL DTRANS_A_B_1(D2M,D2M_f,6561)
1411              CALL DTRANS_A_B_2(Y_new,Y_new_f,3)
1412              CALL DTRANS_A_B_2(Sigma_new,Sigma_new_f,3)
1413              F_pot_new_f = F_pot_new
1414              g_new_f     = g_new
1415 C ***************************************************************
1416            ELSEIF (ID.eq.3) THEN
1417 C ***************************************************************
1418              CALL DTRANS_A_B_2(w_old,w_old_i,3)
1419              CALL DTRANS_A_B_2(w_inv,w_inv_i,3)
1420              CALL DTRANS_A_B_4(M,M_i,3)
1421              CALL DTRANS_A_B_6(DM,DM_i,3)
1422              CALL DTRANS_A_B_1(D2M,D2M_i,6561)
```

```
              CALL DTRANS_A_B_2(Y_new,Y_new_i,3)
              CALL DTRANS_A_B_2(Sigma_new,Sigma_new_i,3)
              F_pot_new_i = F_pot_new
              g_new_i     = g_new
C ====================================================================
            ENDIF
C --------------------------------------------------------------------
C
C       --- Check if all conditions are satisfied
C
            IF (IFlag_damage.eq.0) THEN
C
C             --- There was no damage in the previous increment, hence
C                 stepsize adjustment can be made. If there was damage in
C                 the previous increment the stepsize is fixed for the
C                 current increment to the fixed stepsize in case of
C                 damage. This stepsize is fixed in the variable
C                 dSigma_damage
C
              IF (DABS(g_new - g_old).le.TOL_delta_g) THEN
C
                IFlag_stepsize(ID) = 1
C
              ELSEIF (IFlag_min_step.eq.1) THEN
C
C             --- minimum stepsize has been used in this increment
C
                IFlag_stepsize(ID) = 1
C
              ELSEIF (IFlag_min_step.eq.0) THEN
C
C             --- minimum stepsize is lower then current stepsize
C                 ==> adjust stepsize
C
                IFlag_stepsize(ID) = 0
C
              ENDIF
C
              IFlag_step = IFlag_step + IFlag_stepsize(ID)
C
            ELSEIF (IFlag_damage.eq.1) THEN
C
              IFlag_step = Number_of_Constituents
C
            ELSE
C
              WRITE(*,'(2(A/))')
     +          'An error occurred in the loading',
     +          'The variable IFlag_damagu has a value > 1!!!'
              STOP
C
            ENDIF
C --------------------------------------------------------------------
C
 1110     CONTINUE
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
C
C     --- If conditions are not met entirely take appropriate actions
C
        IF (IFlag_step.lt.Number_of_Constituents)
     +                           STEP_factor = STEP_factor * 0.5d0
        IF (IFlag_step.eq.Number_of_Constituents) THEN
C
          CALL DA_ij_P_B_ij(Sigma_tot,dSigma_tot,Sigma_tot,3)
C
        ENDIF
C --------------------------------------------------------------------
C
 1100   ENDDO
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
        IFlag_damage = 0
C <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
C
        DO 1200 ID = 1, Number_of_Constituents
C
          CALL DINITIALIZE_ZERO_2(dPhi,3)
C
          IFLAG_IMP = 0
          IMP       = 'NN'
C
C       --- Restore the variables as well as the previously calculated
C           values for each constituent
C
```

```
C ***********************************************************************
            IF (ID.eq.1) THEN
C ***********************************************************************
C             --- Matrix material ----------------------------
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_m_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_m_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_m,Eta,3)
              CALL DTRANS_A_B_1(Lambda_m,Lambda,3)
              do i=1,3
                V(i)  = Vi_m(i)
              enddo
              IF (IFlag_restart.eq.0) THEN
                CALL DTRANS_A_B_1(Xi_m_0,Xi,3)
              ELSEIF (IFlag_restart.eq.1) THEN
                CALL DTRANS_A_B_1(Xi_m,Xi,3)
              ENDIF
              c_Gamma   = c_Gamma_m
              F_power   = F_power_m
C
C             --- Variables which change in the process of computation
C
              CALL DTRANS_A_B_2(Sigma_m,Sigma_old,3)
              CALL DTRANS_A_B_2(dSigma_m,dSigma,3)
              CALL DTRANS_A_B_2(Phi_m,Phi_old,3)
              CALL DTRANS_A_B_2(Gamma_m,Gamma_old,3)
              CALL DTRANS_A_B_2(Y_m,Y_old,3)
              Kappa_old = Kappa_m
              g_old     = g_m
              F_pot_old = F_pot_m
C
C             --- Temporary intermediate results
C
              CALL DTRANS_A_B_2(w_old_m,w_old,3)
              CALL DTRANS_A_B_2(w_inv_m,w_inv,3)
              CALL DTRANS_A_B_4(M_m,M,3)
              CALL DTRANS_A_B_6(DM_m,DM,3)
              CALL DTRANS_A_B_1(D2M_m,D2M,6561)
              CALL DTRANS_A_B_2(Y_new_m,Y_new,3)
              CALL DTRANS_A_B_2(Sigma_new_m,Sigma_new,3)
              F_pot_new = F_pot_new_m
              g_new     = g_new_m
C
C ***********************************************************************
            ELSEIF (ID.eq.2) THEN
C ***********************************************************************
C
C             --- Fiber material -----------------------------------
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_f_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_f_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_f,Eta,3)
              CALL DTRANS_A_B_1(Lambda_f,Lambda,3)
              do i=1,3
                V(i)  = Vi_f(i)
              enddo
              IF (IFlag_restart.eq.0) THEN
                CALL DTRANS_A_B_1(Xi_f_0,Xi,3)
              ELSEIF (IFlag_restart.eq.1) THEN
                CALL DTRANS_A_B_1(Xi_f,Xi,3)
              ENDIF
              c_Gamma   = c_Gamma_f
              F_power   = F_power_f
C
C             --- Variables which change in the process of computation
C
              CALL DTRANS_A_B_2(Sigma_f,Sigma_old,3)
              CALL DTRANS_A_B_2(dSigma_f,dSigma,3)
              CALL DTRANS_A_B_2(Phi_f,Phi_old,3)
              CALL DTRANS_A_B_2(Gamma_f,Gamma_old,3)
              CALL DTRANS_A_B_2(Y_f,Y_old,3)
              Kappa_old = Kappa_f
              g_old     = g_f
              F_pot_old = F_pot_f
C
C             --- Temporary intermediate results
C
              CALL DTRANS_A_B_2(w_old_f,w_old,3)
              CALL DTRANS_A_B_2(w_inv_f,w_inv,3)
              CALL DTRANS_A_B_4(M_f,M,3)
```

```
1591              CALL DTRANS_A_B_6(DM_f,DM,3)
1592              CALL DTRANS_A_B_1(D2M_f,D2M,6561)
1593              CALL DTRANS_A_B_2(Y_new_f,Y_new,3)
1594              CALL DTRANS_A_B_2(Sigma_new_f,Sigma_new,3)
1595              F_pot_new = F_pot_new_f
1596              g_new     = g_new_f
1597 C
1598 C ***************************************************************************
1599             ELSEIF (ID.eq.3) THEN
1600 C ***************************************************************************
1601 C
1602 C             --- Interface -------------------------------------
1603 C
1604 C             --- Constants
1605 C
1606              CALL DTRANS_A_B_4(E_i_bar,E_bar,3)
1607              CALL DTRANS_A_B_4(E_i_bar_inv,E_bar_inv,3)
1608              CALL DTRANS_A_B_1(Eta_i,Eta,3)
1609              CALL DTRANS_A_B_1(Lambda_i,Lambda,3)
1610              do i=1,3
1611                V(i)  = Vi_i(i)
1612              enddo
1613              IF (IFlag_restart.eq.0) THEN
1614                CALL DTRANS_A_B_1(Xi_i_0,Xi,3)
1615              ELSEIF (IFlag_restart.eq.1) THEN
1616                CALL DTRANS_A_B_1(Xi_i,Xi,3)
1617              ENDIF
1618              c_Gamma   = c_Gamma_i
1619              F_power   = F_power_i
1620 C
1621 C             --- Variables which change in the process of computation
1622 C
1623              CALL DTRANS_A_B_2(Sigma_i,Sigma_old,3)
1624              CALL DTRANS_A_B_2(dSigma_i,dSigma,3)
1625              CALL DTRANS_A_B_2(Phi_i,Phi_old,3)
1626              CALL DTRANS_A_B_2(Gamma_i,Gamma_old,3)
1627              CALL DTRANS_A_B_2(Y_i,Y_old,3)
1628              Kappa_old = Kappa_i
1629              g_old     = g_i
1630              F_pot_old = F_pot_i
1631 C
1632 C             --- Temporary intermediate results
1633 C
1634                CALL DTRANS_A_B_2(w_old_i,w_old,3)
1635                CALL DTRANS_A_B_2(w_inv_i,w_inv,3)
1636                CALL DTRANS_A_B_4(M_i,M,3)
1637                CALL DTRANS_A_B_6(DM_i,DM,3)
1638                CALL DTRANS_A_B_1(D2M_i,D2M,6561)
1639                CALL DTRANS_A_B_2(Y_new_i,Y_new,3)
1640                CALL DTRANS_A_B_2(Sigma_new_i,Sigma_new,3)
1641                F_pot_new = F_pot_new_i
1642                g_new     = g_new_i
1643 C ***************************************************************************
1644              ENDIF
1645 C ***************************************************************************
1646              CALL DTRANS_A_B_2(dSigma,dSigma_iter,3)
1647 C ---------------------------------------------------------------------------
1648 C             IF (ID.eq.1) write(Check_r,*) ID,'  g_old',g_old
1649 C             IF (ID.eq.1) write(Check_r,*) ID,'  g_new',g_new
1650 C ---------------------------------------------------------------------------
1651 C             Check if new state of stress will produce a damage state
1652 C
1653 C ###########################################################################
1654 C
1655              IF (g_old.lt.0.d0.and.g_new.ge.0.d0) THEN
1656 C
1657 C ###########################################################################
1658 C
1659                IMP = 'IN'
1660                IFlag_damage_constituent(ID) = 1
1661 C
1662                IF (ID.eq.1) I_count = I_count + 1
1663                IF (Incr.eq.0) Incr = L_inc
1664 C
1665 C              Material enters a damage state coming from a non-damaging
1666 C              state
1667 C
1668 C              ---> adjust stress increment such that a stress subincrement
1669 C                   is obtained which will bring the current subincrement
1670 C                   to the damage surface
1671 C
1672 C ---------------------------------------------------------------------------
1673 C                     Improvement according to W. F. Chen - Plasticity
1674 C ---------------------------------------------------------------------------
```

```
1675              exponent  = F_power -1.d0
1676              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
1677 C -----------------------------------------------------------------
1678 C         - Calculate dF_pot_dY_pq
1679 C -----------------------------------------------------------------
1680              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
1681 C -----------------------------------------------------------------
1682 C         - Calculate dY_ij_dSigma_mn
1683 C -----------------------------------------------------------------
1684              CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
1685 C -----------------------------------------------------------------
1686 C         - Calculate dg_dSigma_ij
1687 C -----------------------------------------------------------------
1688              CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
1689              CALL Dc_A_ij(dg_dF_pot,dg_dSigma,3)
1690 C -----------------------------------------------------------------
1691 C         - Calculate the parameter alpha
1692 C -----------------------------------------------------------------
1693              CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha,3)
1694              alpha = - g_old / alpha
1695 C -----------------------------------------------------------------
1696 C         - Calculate the new stress based on parameter alpha
1697 C -----------------------------------------------------------------
1698              CALL DTRANS_A_B_2(dSigma,dum2_1,3)
1699              CALL Dc_A_ij(alpha,dum2_1,3)
1700              CALL DA_ij_P_B_ij(Sigma_old,dum2_1,Sigma_new,3)
1701 C -----------------------------------------------------------------
1702 C         - Check damage surface with new stress
1703 C -----------------------------------------------------------------
1704              CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
1705              F_pot_new = F_pot(w_inv,Y_new,Gamma_old)
1706              g_new     = DXY(F_pot_new,F_power) - 1.d0
1707 C
1708 C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
1709 C
1710 C           R E M A I N D E R   O F   S T R E S S   I N C R E M E N T
1711 C
1712 C         - Now calculate the damage increment for the remainder of
1713 C           the stress increment
1714 C -----------------------------------------------------------------
1715              CALL DTRANS_A_B_2(Sigma_new,Sigma_nold,3)
1716              F_pot_old = F_pot_new
1717              CALL DTRANS_A_B_2(dSigma_iter,dSigma,3)
1718              const = 1.d0 - alpha
1719              CALL Dc_A_ij(const,dSigma,3)
1720              CALL DTRANS_A_B_2(Y_new,Y_old,3)
1721 C -----------------------------------------------------------------
1722 C         - Calculate dg_dF_pot on the damage surface
1723 C -----------------------------------------------------------------
1724              exponent  = F_power -1.d0
1725              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
1726 C -----------------------------------------------------------------
1727 C         - Calculate dF_pot_dY_pq on the damage surface
1728 C -----------------------------------------------------------------
1729              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
1730 C -----------------------------------------------------------------
1731 C         - Calculate dY_ij_dSigma_mn on the damage surface
1732 C -----------------------------------------------------------------
1733              CALL DYDSIGMA(dY_dSigma,Sigma_nold,M,DM,E_bar_inv)
1734 C -----------------------------------------------------------------
1735 C         - Calculate dY_ij_dPhi_mn on the damage surface
1736 C -----------------------------------------------------------------
1737              CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_nold,M,DM,D2M)
1738 C -----------------------------------------------------------------
1739 C         - Calculate dF_pot_dGamma on the damage surface
1740 C -----------------------------------------------------------------
1741              CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
1742 C -----------------------------------------------------------------
1743 C         - Calculate dGamma_dPhi on the damage surface
1744 C -----------------------------------------------------------------
1745              CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
1746 C -----------------------------------------------------------------
1747 C         - Calculate dw_dKappa on the damage surface
1748 C -----------------------------------------------------------------
1749              CALL DINITIALIZE_ZERO_2(dw_dKappa,3)
1750              do i=1, 3
1751                exponent = Xi(i) - 1.d0
1752                dw_dKappa(i,i) = Xi(i) * Eta(i)
1753     +                   * DXY((Kappa_old/Lambda(i)),exponent)
1754              enddo
1755 C -----------------------------------------------------------------
1756 C         - Calculate dF_pot_dw on the damage surface
1757 C -----------------------------------------------------------------
1758              CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
```

```
C ------------------------------------------------------------------
C       - Calculate the new stress level
C ------------------------------------------------------------------
            CALL DA_ij_P_B_ij(Sigma_old,dSigma_iter,Sigma_new,3)
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C       - Calculate the derivatives of g wrt Phi   ---> dg_dPhi
C                                      g wrt Y     ---> dg_dY
C                                      g wrt Gamma ---> dg_dGamma
C                                      g wrt Sigma ---> dg_dSigma
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ------------------------------------------------------------------
C       --- Calculate dg_dSigma
C ------------------------------------------------------------------
            CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
            CALL DC_A_ij(dg_dF_pot,dg_dSigma,3)
C ------------------------------------------------------------------
C       --- Calculate dg_dY on the damage surface
C ------------------------------------------------------------------
            CALL DTRANS_A_B_2(dF_pot_dY,dg_dY,3)
            CALL Dc_A_ij(dg_dF_pot,dg_dY,3)
C ------------------------------------------------------------------
C       --- Calculate dg_dKappa on the damage surface
C ------------------------------------------------------------------
            CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
            dg_dKappa = dg_dF_pot * dg_dKappa
C ------------------------------------------------------------------
C       --- Calculate dg_dPhi on the damage surface
C ------------------------------------------------------------------
            CALL DINITIALIZE_ZERO_2(dum2_1,3)
            CALL DINITIALIZE_ZERO_2(dum2_2,3)
            CALL DINITIALIZE_ZERO_2(dum2_3,3)
C       --- 1. part
            CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C       --- 2. part
            CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
C       --- add the 2 parts
            CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
            CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C       - Calculate Psi as Psi = - Psi_num / Psi_den
C
C       where:    Psi_num_ijkl = dg_dY_ij dg_dSigma_kl
C
C                 Psi_den      = ( dg_dPhi_mn + dg_dKappa * Y_mn
C                                  - c * dg_dY_mn ) * dg_dY_mn
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ------------------------------------------------------------------
C       --- Calculate Psi_den
C ------------------------------------------------------------------
            CALL DTRANS_A_B_2(dg_dY,dum2_1,3)
            CALL Dc_A_ij(c_Gamma,dum2_1,3)
            CALL DTRANS_A_B_2(Y_new,dum2_2,3)
            CALL Dc_A_ij(dg_dKappa,dum2_2,3)
            CALL DA_ij_M_B_ij(dum2_2,dum2_1,dum2_3,3)
            CALL DA_ij_P_B_ij(dum2_3,dg_dPhi,dum2_2,3)
            CALL DA_ij_B_ij(dum2_2,dg_dY,Psi_den,3)
C ------------------------------------------------------------------
C       --- Calculate Psi_num
C ------------------------------------------------------------------
            CALL DA_ij_B_kl(dg_dY,dg_dSigma,Psi_num,3)
C ------------------------------------------------------------------
C       --- Calculate Psi
C ------------------------------------------------------------------
            CALL DTRANS_A_B_4(Psi_num,Psi,3)
            Psi_den_inv = -1.d0 / Psi_den
            CALL Dc_A_ijkl(Psi_den_inv,Psi,3)
C ------------------------------------------------------------------
C       --- Calculate the damage increment dPhi
C ------------------------------------------------------------------
            CALL DA_ijkl_B_kl(Psi,dSigma,dPhi,3)
C ------------------------------------------------------------------
C       --- Update the damage variables
C ------------------------------------------------------------------
C
C       --- Store the old value of Phi
C
            CALL DTRANS_A_B_2(Phi_old,Phi_dum,3)
C
C       --- Update Phi
C
            CALL DA_ij_P_B_ij(Phi_old,dPhi,Phi_new,3)
C ------------------------------------------------------------------
```

```
C         --- Update M, DM, DDM
C -----------------------------------------------------------------
              CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
C -----------------------------------------------------------------
C         --- Update Y
C -----------------------------------------------------------------
              CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C -----------------------------------------------------------------
C         --- Update Kappa
C -----------------------------------------------------------------
              CALL DA_ij_B_ij(Y_old,dPhi,dKappa,3)
              Kappa_new = Kappa_old + dKappa
C -----------------------------------------------------------------
C         --- Calculate w_new and w_inv
C -----------------------------------------------------------------
              do i=1, 3
                do j=1, 3
                  w_new(i,j) = Delta(i,j) * (Lambda(i) * Eta(i) *
     +                         DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i))
                enddo
              enddo
C -----------------------------------------------------------------
              CALL DA_inv_3(w_new,w_inv)
C -----------------------------------------------------------------
C         --- Update Gamma
C -----------------------------------------------------------------
              CALL DTRANS_A_B_2(dPhi,dum2_1,3)
              CALL Dc_A_ij(c_Gamma,dum2_1,3)
              CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
C -----------------------------------------------------------------
C         --- Update F_pot
C -----------------------------------------------------------------
              F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C -----------------------------------------------------------------
C         --- Calculate dY
C -----------------------------------------------------------------
              CALL DA_ij_M_B_ij(Y_new,Y_old,dY,3)
C -----------------------------------------------------------------
C         --- Calculate new g
C -----------------------------------------------------------------
              g_new = DXY(F_pot_new,F_power) - 1.d0
C -----------------------------------------------------------------
C         --- Check for g
C ###################################################################
              IF (dabs(g_new).gt.1.d-4) THEN
C ###################################################################
C
C           ---> Adjust damage variables Phi, Kappa and Gamma such that
C                the damage surface is satisfied using a Taylor series
C                expansion of order 1 based on the following relationships:
C
C                dKappa = Y dPhi
C                dGamma = c dPhi
C -----------------------------------------------------------------
                IMP = '1Y'
C -----------------------------------------------------------------
C           --- Calculate dg_dF_pot
C -----------------------------------------------------------------
                exponent  = F_power - 1.d0
                dg_dF_pot = F_power * DXY(F_pot_old,exponent)
C -----------------------------------------------------------------
C           --- Calculate dF_pot_dY_pq
C -----------------------------------------------------------------
                CALL DA_inv_3(w_old,w_inv)
                CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
C -----------------------------------------------------------------
C           --- Calculate M, DM, DDM based on the variables at the end of
C               the previous step = damage surface is satisfied
C -----------------------------------------------------------------
                CALL M_IJKL_4_81(Phi_old,1,1,1,M,DM,D2M)
C -----------------------------------------------------------------
C           --- Calculate g at the end of the previous step
C -----------------------------------------------------------------
                g_old = DXY(F_pot_old,F_power) - 1.d0
C -----------------------------------------------------------------
C           --- Calculate dg_dSigma at the end of the previous step
C -----------------------------------------------------------------
                CALL DYDSIGMA(dY_dSigma,Sigma_nold,M,DM,E_bar_inv)
                CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dF_pot_dSigma,3)
                CALL Dc_A_ij(dg_dF_pot,dF_pot_dSigma,3)
                CALL DTRANS_A_B_2(dF_pot_dSigma,dg_dSigma,3)
C -----------------------------------------------------------------
C           --- Calculate dg_dKappa at the end of the previous step
C -----------------------------------------------------------------
```

```
1927                CALL DWDKAPPA(dw_dKappa,Xi,Eta,Lambda,Kappa_old)
1928                CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
1929                CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
1930                dg_dKappa = dg_dF_pot * dg_dKappa
1931 C ------------------------------------------------------------------
1932 C           --- Calculate dg_dGamma at the end of the previous step
1933 C                 Note: dg_dGamma = - dg_dY
1934 C ------------------------------------------------------------------
1935                CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
1936                const = -1.d0
1937                CALL Dc_A_ij(const,dF_pot_dY,3)
1938                CALL DTRANS_A_B_2(dF_pot_dY,dg_dGamma,3)
1939                CALL Dc_A_ij(dg_dF_pot,dg_dGamma,3)
1940 C ------------------------------------------------------------------
1941 C           --- Calculate dg_dPhi at the end of the previous step
1942 C ------------------------------------------------------------------
1943                CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
1944                CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
1945                CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_nold,M,DM,D2M)
1946 C           --- 1. part
1947                CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
1948 C           --- 2. part
1949                CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
1950 C           --- add the 2 parts
1951                CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
1952                CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
1953 C ------------------------------------------------------------------
1954 C           --- Calculate alpha as - alpha_num / alpha_den
1955 C ------------------------------------------------------------------
1956                CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha_num,3)
1957                alpha_num = alpha_num + g_old
1958 C
1959                CALL DTRANS_A_B_2(Y_old,dum2_1,3)
1960                CALL Dc_A_ij(dg_dKappa,dum2_1,3)
1961 C
1962                CALL DA_ij_P_B_ij(dg_dPhi,dum2_1,dum2_2,3)
1963 C
1964                CALL DTRANS_A_B_2(dg_dGamma,dum2_1,3)
1965                CALL Dc_A_ij(c_Gamma,dum2_1,3)
1966                CALL DA_ij_P_B_ij(dum2_2,dum2_1,dum2_3,3)
1967                CALL DA_ij_B_ij(dum2_3,dPhi,alpha_den,3)
1968 C
1969                alpha_s = - alpha_num / alpha_den
1970 C ------------------------------------------------------------------
1971 C        --- Now calculate/update the damage affected variables Phi,
1972 C            Kappa and Gamma, and then update the other variables which
1973 C            depend on them
1974 C ------------------------------------------------------------------
1975 C           --- Update Kappa
1976 C
1977                Kappa_new = Kappa_old + alpha_s * dKappa
1978 C
1979 C           --- Update Phi
1980 C
1981                CALL DTRANS_A_B_2(dPhi,dum2_1,3)
1982                CALL Dc_A_ij(alpha_s,dum2_1,3)
1983                CALL DA_ij_P_B_ij(Phi_old,dum2_1,Phi_new,3)
1984 C
1985 C           --- Update Gamma
1986 C
1987                CALL Dc_A_ij(c_Gamma,dum2_1,3)
1988                CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
1989 C
1990 C           --- Calculate w_new and w_inv
1991 C
1992                do i=1, 3
1993                  do j=1, 3
1994                    w_new(i,j) = Delta(i,j) * ( Lambda(i) * Eta(i) *
1995      +                       DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i) )
1996                  enddo
1997                enddo
1998 C
1999                CALL DA_inv_3(w_new,w_inv)
2000 C
2001 C           --- Update M, DM, DDM
2002 C
2003                CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
2004 C
2005 C           --- Update Y
2006 C
2007                CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
2008 C
2009 C           --- Update F_pot
2010 C
```

```
2011                F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
2012 C
2013 C           --- Calculate new g
2014 C
2015                g_new = DXY(F_pot_new,F_power) - 1.d0
2016 C ------------------------------------------------------------------
2017              ENDIF
2018 C ------------------------------------------------------------------
2019 C ------------------------------------------------------------------
2020 C
2021            ELSEIF (g_old.gt.0.d0.and.g_new.gt.0.d0) THEN
2022 C
2023              IMP = '2N'
2024              IFlag_damage_constituent(ID) = 1
2025 C
2026              IF (ID.eq.1) I_count = I_count + 1
2027 C ------------------------------------------------------------------
2028 C
2029 C ------------------------------------------------------------------
2030 C
2031 C           C O M I N G   F R O M   A   D A M A G E   S T A T E
2032 C
2033 C ------------------------------------------------------------------
2034 C
2035 C ------------------------------------------------------------------
2036              CALL DTRANS_A_B_2(dSigma_iter,dSigma,3)
2037 C ------------------------------------------------------------------
2038 C     --- Calculation of dg_dY
2039 C ------------------------------------------------------------------
2040 C         --- Calculate dg_dF_pot
2041 C ------------------------------------------------------------------
2042              exponent  = F_power -1.d0
2043              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
2044 C ------------------------------------------------------------------
2045 C         --- Calculate dF_pot_dY_pq
2046 C ------------------------------------------------------------------
2047              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
2048 C ------------------------------------------------------------------
2049 C         --- Calculate dg_dY
2050 C ------------------------------------------------------------------
2051              CALL DTRANS_A_B_2(dF_pot_dY,dg_dY,3)
2052              CALL Dc_A_ij(dg_dF_pot,dg_dY,3)
2053 C ------------------------------------------------------------------
2054 C         --- Calculate dY_ij_dSigma_mn
2055 C ------------------------------------------------------------------
2056              CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
2057 C ------------------------------------------------------------------
2058 C         --- Calculate dY_ij_dPhi_mn
2059 C ------------------------------------------------------------------
2060              CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_old,M,DM,D2M)
2061 C ------------------------------------------------------------------
2062 C         --- Calculate dF_pot_dGamma
2063 C ------------------------------------------------------------------
2064              CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
2065 C ------------------------------------------------------------------
2066 C         --- Calculate dGamma_dPhi
2067 C ------------------------------------------------------------------
2068              CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
2069 C ------------------------------------------------------------------
2070 C         --- Calculate dw_dKappa
2071 C ------------------------------------------------------------------
2072              CALL DINITIALIZE_ZERO_2(dw_dKappa,3)
2073              DO i=1, 3
2074                exponent = Xi(i) - 1.d0
2075                dw_dKappa(i,i) = Xi(i) * Eta(i)
2076     +                   * DXY((Kappa_old/Lambda(i)),exponent)
2077              ENDDO
2078              exponent = Xi(1) - 1.d0
2079 C ------------------------------------------------------------------
2080 C         --- Calculate dF_pot_dw
2081 C ------------------------------------------------------------------
2082              CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
2083 C ------------------------------------------------------------------
2084 C         --- Calculate the new stress level
2085 C ------------------------------------------------------------------
2086              CALL DA_ij_P_B_ij(Sigma_old,dSigma_iter,Sigma_new,3)
2087 C ------------------------------------------------------------------
2088 C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2089 C       --- Calculate the derivatives of g wrt Phi   ---> dg_dPhi
2090 C                                        g wrt Y     ---> dg_dY
2091 C                                        g wrt Gamma ---> dg_dGamma
2092 C                                        g wrt Sigma ---> dg_dSigma
2093 C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2094 C ------------------------------------------------------------------
```

```
C        --- Calculate dg_dSigma
C -------------------------------------------------------------------
             CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
             CALL DC_A_ij(dg_dF_pot,dg_dSigma,3)
C -------------------------------------------------------------------
C        --- Calculate dg_dY
C -------------------------------------------------------------------
             CALL Dc_A_ij(dg_dF_pot,dF_pot_dY,3)
C -------------------------------------------------------------------
C        --- Calculate dg_dKappa
C -------------------------------------------------------------------
             CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
             dg_dKappa = dg_dF_pot * dg_dKappa
C -------------------------------------------------------------------
C        --- Calculate dg_dPhi on the damage surface
C -------------------------------------------------------------------
             CALL DINITIALIZE_ZERO_2(dum2_1,3)
             CALL DINITIALIZE_ZERO_2(dum2_2,3)
             CALL DINITIALIZE_ZERO_2(dum2_3,3)
C        --- 1. part
             CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C        --- 2. part
             CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
C        --- add the 2 parts
             CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
             CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C        - Calculate Psi as Psi = - Psi_num / Psi_den
C
C        where:    Psi_num_ijkl = dg_dY_ij dg_dSigma_kl
C
C                  Psi_den      = ( dg_dPhi_mn + dg_dKappa * Y_mn
C                                   - c * dg_dY_mn ) * dg_dY_mn
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C -------------------------------------------------------------------
C        --- Calculate Psi_den
C -------------------------------------------------------------------
             CALL DTRANS_A_B_2(dg_dY,dum2_1,3)
             CALL Dc_A_ij(c_Gamma,dum2_1,3)
             CALL DTRANS_A_B_2(Y_new,dum2_2,3)
             CALL Dc_A_ij(dg_dKappa,dum2_2,3)
             CALL DA_ij_M_B_ij(dum2_2,dum2_1,dum2_3,3)
             CALL DA_ij_P_B_ij(dum2_3,dg_dPhi,dum2_2,3)
             CALL DA_ij_B_ij(dum2_2,dg_dY,Psi_den,3)
C -------------------------------------------------------------------
C        --- Calculate Psi_num
C -------------------------------------------------------------------
             CALL DA_ij_B_kl(dg_dY,dg_dSigma,Psi_num,3)
C -------------------------------------------------------------------
C        --- Calculate Psi
C -------------------------------------------------------------------
             CALL DTRANS_A_B_4(Psi_num,Psi,3)
             Psi_den_inv = -1.d0 / Psi_den
             CALL Dc_A_ijkl(Psi_den_inv,Psi,3)
C -------------------------------------------------------------------
C        --- Calculate the damage increment dPhi
C -------------------------------------------------------------------
             CALL DA_ijkl_B_kl(Psi,dSigma,dPhi,3)
C -------------------------------------------------------------------
C        - Update the damage variables
C -------------------------------------------------------------------
C
C        --- Store the old value of Phi
C
             CALL DTRANS_A_B_2(Phi_old,Phi_dum,3)
C
C        --- Update Phi
C
             CALL DA_ij_P_B_ij(Phi_old,dPhi,Phi_new,3)
C
C        --- Update M, DM, DDM
C
             CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
C
C        --- Update Y
C
             CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C
C        --- Update Kappa
C
             CALL DA_ij_B_ij(Y_old,dPhi,dKappa,3)
             Kappa_new = Kappa_old + dKappa
```

```
C
C       --- Calculate w_new and w_inv
C
            do i=1, 3
              do j=1, 3
                w_new(i,j) = Delta(i,j) * (Lambda(i) * Eta(i) *
     +                       DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i) )
              enddo
            enddo
            CALL DA_inv_3(w_new,w_inv)
C
C       --- Update Gamma
C
            CALL DTRANS_A_B_2(dPhi,dum2_1,3)
            CALL Dc_A_ij(c_Gamma,dum2_1,3)
            CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
C
C       --- Update F_pot
C
            F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C
C       --- Calculate dY
C
            CALL DA_ij_M_B_ij(Y_new,Y_old,dY,3)
C
C       --- Calculate new g
C
            g_new = DXY(F_pot_new,F_power) - 1.d0
C
C ------------------------------------------------------------------
            IF (DABS(g_new).gt.1.d-4) THEN
C ------------------------------------------------------------------
C
C         ---> Adjust damage variables Phi, Kappa and Gamma such that
C              the damage surface is satisfied using a Taylor series
C              expansion of order 1 based on the following relationships:
C
C              dKappa = Y dPhi
C              dGamma = c dPhi
C
               IMP = '2Y'
C ------------------------------------------------------------------
C       --- Calculate dg_dF_pot
C ------------------------------------------------------------------
               exponent  = F_power -1.d0
               dg_dF_pot = F_power * DXY(F_pot_old,exponent)
C ------------------------------------------------------------------
C       --- Calculate dF_pot_dY_pq
C ------------------------------------------------------------------
               CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
C ------------------------------------------------------------------
C       --- Calculate M, DM, DDM based on the variables at the end of
C           the previous step = damage surface is satisfied
C ------------------------------------------------------------------
               CALL M_IJKL_4_81(Phi_old,1,1,1,M,DM,D2M)
               CALL DA_inv_3(w_old,w_inv)
C ------------------------------------------------------------------
C       --- Calculate g at the end of the previous step
C ------------------------------------------------------------------
               g_old = DXY(F_pot_old,F_power) - 1.d0
C ------------------------------------------------------------------
C       --- Calculate dg_dSigma at the end of the previous step
C ------------------------------------------------------------------
               CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
               CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dF_pot_dSigma,3)
               CALL Dc_A_ij(dg_dF_pot,dF_pot_dSigma,3)
               CALL DTRANS_A_B_2(dF_pot_dSigma,dg_dSigma,3)
C ------------------------------------------------------------------
C       --- Calculate dg_dKappa at the end of the previous step
C ------------------------------------------------------------------
               CALL DWDKAPPA(dw_dKappa,Xi,Eta,Lambda,Kappa_old)
               CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
               CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
               dg_dKappa = dg_dF_pot * dg_dKappa
C ------------------------------------------------------------------
C       --- Calculate dg_dPhi at the end of the previous step
C ------------------------------------------------------------------
               CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
               CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
               CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_old,M,DM,D2M)
C       --- 1.part
               CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C       --- 2.part
               CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
```

```
C          --- add the 2 parts
               CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
               CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C -----------------------------------------------------------------
C          --- Calculate dg_dGamma at the end of the previous step
C              Note: dg_dGamma = - dg_dY
C -----------------------------------------------------------------
               CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
               const = -1.d0
               CALL Dc_A_ij(const,dF_pot_dY,3)
               CALL DTRANS_A_B_2(dF_pot_dY,dg_dGamma,3)
               CALL Dc_A_ij(dg_dF_pot,dg_dGamma,3)
C -----------------------------------------------------------------
C          --- Calculate alpha = - alpha_num / alpha_den
C -----------------------------------------------------------------
               CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha_num,3)
               alpha_num = alpha_num + g_old
C
               CALL DTRANS_A_B_2(Y_old,dum2_1,3)
               CALL Dc_A_ij(dg_dKappa,dum2_1,3)
               CALL DA_ij_P_B_ij(dg_dPhi,dum2_1,dum2_2,3)
               CALL DTRANS_A_B_2(dg_dGamma,dum2_1,3)
               CALL Dc_A_ij(c_Gamma,dum2_1,3)
               CALL DA_ij_P_B_ij(dum2_2,dum2_1,dum2_3,3)
               CALL DA_ij_B_ij(dum2_3,dPhi,alpha_den,3)
C
               alpha_s = - alpha_num / alpha_den
C -----------------------------------------------------------------
C       --- Now calculate/update the damage affected variables Phi,
C           Kappa, Gamma
C -----------------------------------------------------------------
C
C          --- Update Kappa
C
               Kappa_new = Kappa_old + alpha_s * dKappa
C
C          --- Update Phi
C
               CALL DTRANS_A_B_2(dPhi,dum2_1,3)
               CALL Dc_A_ij(alpha_s,dum2_1,3)
               CALL DA_ij_P_B_ij(Phi_old,dum2_1,Phi_new,3)
C
```

```
C          --- Update Gamma
C
               CALL Dc_A_ij(c_Gamma,dum2_1,3)
               CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
C
C          --- Calculate w_new and w_inv
C
               do i=1, 3
                 do j=1, 3
                   w_new(i,j) = Delta(i,j) + ( Lambda(i) * Eta(i) *
     +                    DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i) )
                 enddo
               enddo
               CALL DA_inv_3(w_new,w_inv)
C
C          --- Update M, DM, DDM
C
               CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
C
C          --- Update Y
C
               CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C
C          --- Update F_pot
C
               F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C
C          --- Calculate new g
C
               g_new = DXY(F_pot_new,F_power) - 1.d0
C
             ENDIF
C -----------------------------------------------------------------
C -----------------------------------------------------------------
C
            ELSEIF ((g_old.gt.0.d0.and.g_new.le.0.d0).or.
     +                    (g_old.le.0.d0.and.g_new.le.0.d0)) THEN
C
C -----------------------------------------------------------------
C
C            No damage neither previous increment nor in the current
C
```

```
2347              IFlag_damage_constituent(ID) = 0
2348 C
2349              CALL DTRANS_A_B_2(Phi_old,Phi_new,3)
2350              Kappa_new = Kappa_old
2351 C
2352              CALL DTRANS_A_B_2(Gamma_old,Gamma_new,3)
2353 C
2354            ENDIF
2355 C
2356 C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
2357 C
2358 C         Store Variables in the appropriate places for each constituent
2359 C
2360 C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
2361 C ---------------------------------------------------------------------
2362            IF (ID.eq.1) THEN
2363 C ---------------------------------------------------------------------
2364 C
2365 C            Matrix material
2366 C
2367              CALL DTRANS_A_B_2(Sigma_new,Sigma_m,3)
2368              CALL DTRANS_A_B_2(Phi_new,Phi_m,3)
2369              CALL DTRANS_A_B_2(Gamma_new,Gamma_m,3)
2370              CALL DTRANS_A_B_2(Y_new,Y_m,3)
2371              CALL DTRANS_A_B_4(M,M_m,3)
2372              Kappa_m = Kappa_new
2373              g_m     = g_new
2374              F_pot_m = F_pot_new
2375              If (Kappa_m_0.lt.1.d-15) Kappa_m_0 = Kappa_new
2376 C ---------------------------------------------------------------------
2377 C           Printout to check
2378 C
2379            write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A2,1x,
2380     +                          d12.6,2x,f6.3,2x,f8.5,3x,f10.5)')
2381     +         L_inc,g_m,Y_m(1,1),F_pot_m,Phi_m(1,1),Xi(1),
2382     +         Sigma_m(1,1),IMP,Kappa_m,c_m_adj,w_old(1,1),
2383     +         Sigma_tot(1,1)
2384 C
2385 C ---------------------------------------------------------------------
2386            ELSEIF (ID.eq.2) THEN
2387 C ---------------------------------------------------------------------
2388 C
2389 C            Fiber material
2390 C
2391              CALL DTRANS_A_B_2(Sigma_new,Sigma_f,3)
2392              CALL DTRANS_A_B_2(Phi_new,Phi_f,3)
2393              CALL DTRANS_A_B_2(Y_new,Y_f,3)
2394              CALL DTRANS_A_B_2(Gamma_new,Gamma_f,3)
2395              CALL DTRANS_A_B_4(M,M_f,3)
2396              Kappa_f = Kappa_new
2397              g_f     = g_new
2398              F_pot_f = F_pot_new
2399              If (Kappa_f_0.lt.1.d-15) Kappa_f_0 = Kappa_new
2400 C ---------------------------------------------------------------------
2401 C           Printout to check
2402 C
2403              write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A,1x,
2404     +                                    d12.6,2x,f6.3,2x,f8.5)')
2405     +          L_inc,g_f,Y_f(1,1),F_pot_f,Phi_f(1,1),Xi(1),
2406     +          Sigma_f(1,1),IMP,Kappa_f,c_f_adj,w_old(1,1)
2407 C ---------------------------------------------------------------------
2408            ELSEIF (ID.eq.3) THEN
2409 C ---------------------------------------------------------------------
2410 C
2411 C            ---- Interface ----------------
2412 C
2413              CALL DTRANS_A_B_2(Sigma_new,Sigma_i,3)
2414              CALL DTRANS_A_B_2(Phi_new,Phi_i,3)
2415              CALL DTRANS_A_B_2(Y_new,Y_i,3)
2416              CALL DTRANS_A_B_2(Gamma_new,Gamma_i,3)
2417              CALL DTRANS_A_B_4(M,M_i,3)
2418              Kappa_i = Kappa_new
2419              g_i     = g_new
2420              F_pot_i = F_pot_new
2421              If (Kappa_m_0.lt.1.d-19) Kappa_m_0 = Kappa_new
2422 C ---------------------------------------------------------------------
2423 C           Printout to check
2424 C
2425              write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A,1x,
2426     +                                    d12.6,2x,f6.3,2x,f8.5)')
2427     +          L_inc,g_i,Y_i(1,1),F_pot_i,Phi_i(1,1),Xi(1),
2428     +          Sigma_i(1,1),IMP,Kappa_i,w_old(1,1)
2429 C
2430 C ---------------------------------------------------------------------
```

```
            ENDIF
C ***************************************************************************
C   --- Check whether damage occurs and set the flag for it in order to
C       set the stress increment properly.
C
            IF (Iflag_damage_constituent(ID).eq.1) IFlag_damage = 1
C
C ------- End of Loop on constituents ---------------------------------
C
1200      CONTINUE
C
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C ------- Overall damage in the composite -----------------------------
C
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
            CALL COMPOSITE_DAMAGE(c_f,c_m,Phi_f,Phi_m,Phi_c)
C
C ------- Ouput for checking purposes ---------------------------------
C
          IF (Number_of_Cycles.le.4)
     +      CALL STRESS_STRAIN_DATA(L_inc,Sigma_m,Sigma_f,
     +                E_m_bar_inv,E_f_bar_inv,M_m,M_f,
     +                Epsilon_m,Epsilon_f,SS_dat)
C
C ----- End of increment loop
C
1000    CONTINUE
C
      ENDIF
C
C ###########################################################################
C
C                   C y c l i c   L o a d i n g
C
C ###########################################################################
C
C ---------------------------------------------------------------------------
C     L_Step = Load step counter
C     N_inc  = number of load increments
C ---------------------------------------------------------------------------
C     Distinguish wave forms
C
      IF (IFlag_L.EQ.1) THEN
C
C     --- Sinusoidal wave form ---
C
C     --- Set the stepsize in the damage region in dSigma_damage
C         The coefficient stepsize controls the stepsize
C         since the loop only determines the maximum component of the
C         cyclic amplitude and then divides the stress amplitude by
C         the absolute max. value.
C
        factor        = Damage_stepsize_cyclic / 180.d0
        dTheta_damage = factor * Pi
C
        dTheta_0 = Pi / (2.d0 * Number_of_discretization_points)
C
C -----------------------------------------------------------------------
      ELSEIF (IFlag_L.EQ.2) THEN
C
C     --- Triangular wave form ---
C
        DO i=1,3
          DO j=i,3
            dSigma_tot(i,j) = Sigma_Amp(i,j) / N_inc
            IF (i.NE.j) dSigma_tot(j,i) = dSigma_tot(i,j)
          ENDDO
        ENDDO
C
      ELSEIF (IFlag_L.EQ.4) THEN
C
C     --- Monotonic Loading---
C
C
      ENDIF
C -----------------------------------------------------------------------
C
C     --- Start of Loading
C
C     --- Current stress is mean stress
C ***********************************************************************
      CALL DTRANS_A_B_2(Sigma_mean,Sigma_tot,3)
```

```
C ------------------------------------------------------------------
C ------------------------------------------------------------------
C
C ---------------------- START OF CYCLIC LOADING ---------------------
C
C ------------------------------------------------------------------
      IF (IFlag_Restart.eq.1) THEN
        write(check_r,'(A/)') 'Restart with no increase in mean stress'
      ENDIF
C ------------------------------------------------------------------
C
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      DO 2000 NOC=1, Number_of_Cycles
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C     --- Set IFlag_Xi in order to indicate that Xi has to be adjusted
C         for this cycle
C
        Theta_new = 0.d0
        Theta_old = 0.d0
        IFlag_Xi  = 1
C ------------------------------------------------------------------
C
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        DO 2100 WHILE((2.d0*Pi-Theta_new).gt.1.d-5)
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C ------------------------------------------------------------------
C
          L_inc = L_inc + 1
C ------------------------------------------------------------------
          STEP_factor     = 1.d0
          IFlag_min_step  = 0
C
          IF (IFlag_L.EQ.1) THEN
C ------------------------------------------------------------------
C     --- Sinusoidal load
C
C         The increment in load is determined by taking the difference
C         in the sine of the angles befors and after the load increment
C         which is basically done as
C
C         D_Sigma = ( SIN(Theta_new) - SIN(Theta_old) ) * Delta_Sigma
C
C         where Delta_Sigma represents the stress amplitude of the
C         cyclic loading.
C ------------------------------------------------------------------
C       --- Identify the current quarter of the load cycle
C
            IF (Theta_old.lt.0.5d0*Pi) Load_quarter = 1
            IF (Theta_old.ge.0.5d0*Pi.and.Theta_old.lt.Pi)
     +         Load_quarter = 2
            IF (Theta_old.ge.Pi.and.Theta_old.lt.1.5d0*Pi)
     +         Load_quarter = 3
            IF (Theta_old.ge.1.5d0*Pi.and.Theta_old.lt.2.d0*Pi)
     +         Load_quarter = 4
C ##################################################################
            IF (IFlag_damage.eq.0) THEN
C ##################################################################
C         --- No damage occurred in the previous increment
C
C
              IFlag_min_step = 0
C ------------------------------------------------------------------
              IF (Load_quarter.eq.1) THEN
C ------------------------------------------------------------------
                IF ((0.5d0*Pi-Theta_old).gt.dTheta_0) THEN
C
                  dTheta = dTheta_0
C
                ELSE
C
                  dTheta = 0.5d0*Pi - Theta_old
C
                ENDIF
C ------------------------------------------------------------------
              ELSEIF (Load_quarter.eq.2) THEN
C ------------------------------------------------------------------
                IF ((Pi-Theta_old).gt.dTheta_0) THEN
C
                  dTheta = dTheta_0
C
                ELSE
C
```

```
2599                dTheta = Pi - Theta_old
2600 C
2601              ENDIF
2602 C -------------------------------------------------------------------
2603            ELSEIF (Load_quarter.eq.3) THEN
2604 C -------------------------------------------------------------------
2605              IF ((1.5d0*Pi-Theta_old).gt.dTheta_0) THEN
2606 C
2607                dTheta = dTheta_0
2608 C
2609              ELSE
2610 C
2611                dTheta = 1.5d0*Pi - Theta_old
2612 C
2613              ENDIF
2614 C -------------------------------------------------------------------
2615            ELSEIF (Load_quarter.eq.4) THEN
2616 C -------------------------------------------------------------------
2617              IF ((2.d0*Pi-Theta_old).gt.dTheta_0) THEN
2618 C
2619                dTheta = dTheta_0
2620 C
2621              ELSE
2622 C
2623                dTheta = 2.d0*Pi - Theta_old
2624 C
2625              ENDIF
2626 C -------------------------------------------------------------------
2627            ENDIF
2628 C ###################################################################
2629          ELSEIF (IFlag_damage.eq.1) THEN
2630 C ###################################################################
2631 C       --- Damage occurred in the previous increment
2632 C           ---> set the stepsize to the minimum for damage or to the
2633 C                difference between the region limit and the current
2634 C                stress level is the difference happens to be lower than
2635 C                the minimum stepsize.
2636 C           ---> In all cases the flag for the minimum stepsize must be
2637 C                set to 1
2638 C -------------------------------------------------------------------
2639              IFlag_min_step = 1
2640 C -------------------------------------------------------------------
2641            IF (Load_quarter.eq.1) THEN
2642 C
2643              IF ((0.5d0*Pi-Theta_old).gt.dTheta_damage) THEN
2644 C
2645                dTheta = dTheta_damage
2646 C
2647              ELSE
2648 C
2649                dTheta = 0.5d0*Pi - Theta_old
2650 C
2651              ENDIF
2652 C -------------------------------------------------------------------
2653            ELSEIF (Load_quarter.eq.2) THEN
2654 C -------------------------------------------------------------------
2655              IF ((Pi-Theta_old).gt.dTheta_damage) THEN
2656 C
2657                dTheta = dTheta_damage
2658 C
2659              ELSE
2660 C
2661                dTheta = Pi - Theta_old
2662 C
2663              ENDIF
2664 C -------------------------------------------------------------------
2665            ELSEIF (Load_quarter.eq.3) THEN
2666 C -------------------------------------------------------------------
2667              IF ((1.5d0*Pi-Theta_old).gt.dTheta_damage) THEN
2668 C
2669                dTheta = dTheta_damage
2670 C
2671              ELSE
2672 C
2673                dTheta = 1.5d0*Pi - Theta_old
2674 C
2675              ENDIF
2676 C -------------------------------------------------------------------
2677            ELSEIF (Load_quarter.eq.4) THEN
2678 C -------------------------------------------------------------------
2679              IF ((2.d0*Pi-Theta_old).gt.dTheta_damage) THEN
2680 C
2681                dTheta = dTheta_damage
2682 C
```

```
2683                ELSE
2684 C
2685                  dTheta = 2.d0*Pi - Theta_old
2686 C
2687                ENDIF
2688 C ---------------------------------------------------------------------
2689              ENDIF
2690 C #####################################################################
2691            ENDIF
2692 C #####################################################################
2693 C ---------------------------------------------------------------------
2694          ELSEIF (IFlag_L.EQ.2) THEN
2695 C
2696 C     --- Triangular wave form ----------------------------------------
2697 C
2698 C        Here 3 regions are to be distinguished for the loading:
2699 C
2700 C
2701 C                /\          /\          /\
2702 C               /  \        /  \        /  \
2703 C              /    \      /    \      /    \      mean stress
2704 C           ---------\----/------\----/------\-----------------
2705 C                     \  /        \  /        \  /
2706 C              |  |    \/ |        \/          \/
2707 C              |  |       |
2708 C              |  |       |
2709 C              |  |     |  |
2710 C              |  |     |  |
2711 C               1    2    3
2712 C
2713 C              |           |
2714 C              |  1 cycle  |
2715 C              -------------
2716 C
2717 C     ---------------------------------------------------------------
2718            DO i=1,3
2719              DO j=i,3
2720 C
2721                IF (L_step.le.N_inc+1) THEN
2722 C
2723 C              --- Region 1 ---
2724 C
2725                  dSigma_tot(i,j) = dSigma_tot(i,j)
2726 C
2727                ELSEIF (L_Step.gt.N_inc+1.and.L_Step.le.3*N_inc+1) THEN
2728 C
2729 C              --- Region 2 ---
2730 C
2731                  dSigma_tot(i,j) = - dabs(dSigma_tot(i,j))
2732 C
2733                  ELSE
2734 C
2735 C              --- Region 3 ---
2736 C
2737                  dSigma_tot(i,j) = + dSigma_tot(i,j)
2738 C
2739                ENDIF
2740 C
2741                IF (i.NE.j) Sigma_tot(j,i) = Sigma_tot(i,j)
2742 C
2743              ENDDO
2744            ENDDO
2745 C
2746          ENDIF
2747 C
2748 C -------------------------------------------------------------------
2749 C     --- Calculate Effective volume fractions
2750 C -------------------------------------------------------------------
2751          CALL EFF_VOL_FRAC(c_f,c_m,c_f_0,c_m_0,Phi_f,Phi_m,
2752     +                                          Phi_f_cr,Phi_m_cr)
2753 C
2754          c_m_adj = c_m
2755          c_f_adj = c_f
2756 C
2757          IF (IFlag_EVF.eq.0) THEN
2758            c_m     = c_m_0
2759            c_f     = c_f_0
2760          ENDIF
2761 C -------------------------------------------------------------------
2762 C     --- Calculate stress and strain concentration tensors:
2763 C -------------------------------------------------------------------
2764          CALL M_ijkl_4_81(Phi_m,1,0,0,M_m,DM,D2M)
2765          CALL M_ijkl_4_81(Phi_f,1,0,0,M_f,DM,D2M)
2766          CALL SSCF(Nu_m,c_f,E_f_bar,E_f_bar_inv,E_m_bar,E_m_bar_inv,S,
```

```
     +          M_m,M_f,A_f_bar,B_f_bar,A_m_bar,B_m_bar,A_f,B_f,A_m,B_m)
C ------------------------------------------------------------------
C     Note: The stress has to be amplified by M in order to account for the
C           reduction in effective area/volume
C           ---> calculate M of the composite to adjust the stress before
C                distributing it!!!
C ------------------------------------------------------------------
          CALL M_COMPOSITE(c_f,c_m,M_f,M_m,B_f,B_m,M_comp)
C
C         --- Calculate effective applied stress
C
          CALL DA_ijkl_B_kl(M_comp,Sigma_tot,Sigma_tot_eff,3)
C ------------------------------------------------------------------
C     --- Calculate the stresses and stress increments in
C         the constituents
C ------------------------------------------------------------------
          IF (IFlag_effstress.eq.1) THEN
C
            CALL DA_ijkl_B_kl(B_m_bar,Sigma_tot_eff,Sigma_m,3)
            CALL DA_ijkl_B_kl(B_f_bar,Sigma_tot_eff,Sigma_f,3)
C
          ELSEIF (IFlag_effstress.eq.0) THEN
C
            CALL DA_ijkl_B_kl(B_m,Sigma_tot_eff,Sigma_m,3)
            CALL DA_ijkl_B_kl(B_f,Sigma_tot_eff,Sigma_f,3)
C
          ENDIF
C ------------------------------------------------------------------
C     --- Calculate the stress on the interface as the difference of the
C         stresses in the constituents
C ------------------------------------------------------------------
          IF (IFlag_Interface.eq.1) THEN
            CALL DA_ij_M_B_ij(Sigma_f,Sigma_m,Sigma_i,3)
          ENDIF
C ------------------------------------------------------------------
C          CALL DA_ij_P_B_ij(Sigma_tot,dSigma_tot,Sigma_tot,3)
C ------------------------------------------------------------------
          N_it             = 0
          IFlag_step       = 0
          Reduction_factor = 1.d0
          CALL I_INITIALIZE_ZERO_1(IFlag_stepsize,3)
C ------------------------------------------------------------------
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          DO 2200 WHILE(IFlag_step.lt.Number_of_Constituents)
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C ------------------------------------------------------------------
            IFlag_step = 0
            N_it = N_it + 1
C
            IF (IFlag_L.eq.1) THEN
C
C         --- Sinusoidal load
C
              dTheta  = STEP_factor * dTheta
              IF (dTheta.lt.dTheta_damage.and.IFlag_min_step.eq.0) THEN
C
                dTheta = dTheta_damage
                IFlag_min_step = 1
C
              ENDIF
C
              Theta_New  = Theta_old + dTheta
              dSin_Theta = DSIN(Theta_New) - DSIN(Theta_Old)
C ------------------------------------------------------------------
              DO i=1,3
                DO j=i,3
                  dSigma_tot(i,j) = dSin_Theta * Sigma_Amp(i,j)
                  IF (i.NE.j) dSigma_tot(j,i) = dSigma_tot(i,j)
                ENDDO
              ENDDO
C
            ENDIF
C ------------------------------------------------------------------
C
C         --- Calculate effective applied stress increment
C
            CALL DA_ijkl_B_kl(M_comp,dSigma_tot,dSigma_tot_eff,3)
C ------------------------------------------------------------------
C       --- Calculate the stress increments in the constituents
C ------------------------------------------------------------------
C
            IF (IFlag_effstress.eq.1) THEN
C
              CALL DA_ijkl_B_kl(B_m_bar,dSigma_tot_eff,dSigma_m,3)
```

```
2851              CALL DA_ijkl_B_kl(B_f_bar,dSigma_tot_eff,dSigma_f,3)
2852 C
2853            ELSEIF (IFlag_effstress.eq.0) THEN
2854 C
2855              CALL DA_ijkl_B_kl(B_m,dSigma_tot_eff,dSigma_m,3)
2856              CALL DA_ijkl_B_kl(B_f,dSigma_tot_eff,dSigma_f,3)
2857 C
2858            ENDIF
2859 C -------------------------------------------------------------------
2860 C     --- Calculate the stress increment on the interface as the
2861 C         difference of the stresses in the constituents
2862 C -------------------------------------------------------------------
2863           IF (IFlag_Interface.eq.1) THEN
2864             CALL DA_ij_M_B_ij(dSigma_f,dSigma_m,dSigma_i,3)
2865           ENDIF
2866 C -------------------------------------------------------------------
2867 C
2868             CALL DINITIALIZE_ZERO_2(dPhi,3)
2869 C
2870 C -------------------------------------------------------------------
2871 C       --- Loop over the number of constituents
2872 C -------------------------------------------------------------------
2873 C             ID =1  ---> matrix
2874 C             ID =2  ---> fiber
2875 C             ID =3  ---> interface
2876 C -------------------------------------------------------------------
2877 C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
2878             DO 2210 ID = 1, Number_of_Constituents
2879 C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
2880 C -------------------------------------------------------------------
2881 C       --- Transfer constituent variables to global variables
2882 C ===================================================================
2883               IF (ID.eq.1) THEN
2884 C ===================================================================
2885 C
2886 C           --- Matrix material
2887 C
2888 C           --- Constants
2889 C
2890                 CALL DTRANS_A_B_4(E_m_bar,E_bar,3)
2891                 CALL DTRANS_A_B_4(E_m_bar_inv,E_bar_inv,3)
2892                 CALL DTRANS_A_B_1(Eta_m,Eta,3)
2893                 CALL DTRANS_A_B_1(Lambda_m,Lambda,3)
2894                 do i=1,3
2895                   V(i)  = Vi_m(i)
2896                 enddo
2897 C
2898                 c_Gamma   = c_Gamma_m
2899                 F_power   = F_power_m
2900 C
2901 C           --- Variables which change in the process of computation
2902 C
2903                 CALL DTRANS_A_B_2(Sigma_m,Sigma_old,3)
2904                 CALL DTRANS_A_B_2(dSigma_m,dSigma,3)
2905                 CALL DTRANS_A_B_2(Phi_m,Phi_old,3)
2906                 CALL DTRANS_A_B_2(Gamma_m,Gamma_old,3)
2907                 CALL DTRANS_A_B_2(Y_m,Y_old,3)
2908                 Kappa_old = Kappa_m
2909                 g_old     = g_m
2910                 F_pot_old = F_pot_m
2911 C
2912                 IF (IFlag_Xi.eq.1)
2913      +            CALL ADJUST_XI(Xi_m_0,Xi_m,dXi_m1,dXi_m2,NOC,ID,N_m1,
2914      +                           N_m2,IFlag_Restart,Number_of_Cycles_old,
2915      +                           a_Matrix,a_Fiber,a_Interface)
2916 C
2917                 CALL DTRANS_A_B_1(Xi_m,Xi,3)
2918 C ===================================================================
2919               ELSEIF (ID.eq.2) THEN
2920 C ===================================================================
2921 C
2922 C           --- Fiber material
2923 C
2924 C           --- Constants
2925 C
2926                 CALL DTRANS_A_B_4(E_f_bar,E_bar,3)
2927                 CALL DTRANS_A_B_4(E_f_bar_inv,E_bar_inv,3)
2928                 CALL DTRANS_A_B_1(Eta_f,Eta,3)
2929                 CALL DTRANS_A_B_1(Lambda_f,Lambda,3)
2930                 do i=1,3
2931                   V(i)  = Vi_f(i)
2932                 enddo
2933 C
2934                 c_Gamma   = c_Gamma_f
```

```
2935                F_power   = F_power_f
2936 C
2937 C           --- Variables which change in the process of computation
2938 C
2939                CALL DTRANS_A_B_2(Sigma_f,Sigma_old,3)
2940                CALL DTRANS_A_B_2(dSigma_f,dSigma,3)
2941                CALL DTRANS_A_B_2(Phi_f,Phi_old,3)
2942                CALL DTRANS_A_B_2(Gamma_f,Gamma_old,3)
2943                CALL DTRANS_A_B_2(Y_f,Y_old,3)
2944                Kappa_old = Kappa_f
2945                g_old     = g_f
2946                F_pot_old = F_pot_f
2947 C
2948                IF (IFlag_Xi.eq.1)
2949      +           CALL ADJUST_XI(Xi_f_0,Xi_f,dXi_f1,dXi_f2,NOC,ID,N_f1,
2950      +                          N_f2,IFlag_Restart,Number_of_Cycles_old,
2951      +                          a_Matrix,a_Fiber,a_Interface)
2952 C
2953                CALL DTRANS_A_B_1(Xi_f,Xi,3)
2954 C ***************************************************************************
2955              ELSEIF (ID.eq.3) THEN
2956 C ***************************************************************************
2957 C
2958 C           --- Interface material
2959 C
2960 C           --- Constants
2961 C
2962                CALL DTRANS_A_B_4(E_i_bar,E_bar,3)
2963                CALL DTRANS_A_B_4(E_i_bar_inv,E_bar_inv,3)
2964                CALL DTRANS_A_B_1(Eta_i,Eta,3)
2965                CALL DTRANS_A_B_1(Lambda_i,Lambda,3)
2966                do i=1,3
2967                  V(i)  = Vi_i(i)
2968                enddo
2969 C
2970                c_Gamma   = c_Gamma_i
2971                F_power   = F_power_i
2972 C
2973 C           --- Variables which change in the process of computation
2974 C
2975                CALL DTRANS_A_B_2(Sigma_i,Sigma_old,3)
2976                CALL DTRANS_A_B_2(dSigma_i,dSigma,3)
2977                CALL DTRANS_A_B_2(Phi_i,Phi_old,3)
2978                CALL DTRANS_A_B_2(Gamma_i,Gamma_old,3)
2979                CALL DTRANS_A_B_2(Y_i,Y_old,3)
2980                Kappa_old = Kappa_i
2981                g_old     = g_i
2982                F_pot_old = F_pot_i
2983 C
2984                IF (IFlag_Xi.eq.1)
2985      +           CALL ADJUST_XI(Xi_i_0,Xi_i,dXi_i1,dXi_i2,NOC,ID,N_i1,
2986      +                          N_i2,IFlag_Restart,Number_of_Cycles_old,
2987      +                          a_Matrix,a_Fiber,a_Interface)
2988 C
2989                CALL DTRANS_A_B_1(Xi_i,Xi,3)
2990 C ***************************************************************************
2991              ENDIF
2992 C ***************************************************************************
2993 C           - Reset IFlag_Xi for next loading cycle since only one
2994 C             adjustment is done per cycle
2995 C ---------------------------------------------------------------------------
2996                IF (ID.eq.Number_of_Constituents.and.IFlag_Xi.eq.1)
2997      +           IFlag_Xi = 0
2998 C ---------------------------------------------------------------------------
2999 C           - Calculate new stress level and store dSigma for each
3000 C             iteration
3001 C ---------------------------------------------------------------------------
3002                CALL DTRANS_A_B_2(dSigma,dSigma_iter,3)
3003                CALL DA_ij_P_B_ij(Sigma_old,dSigma,Sigma_new,3)
3004 C ---------------------------------------------------------------------------
3005 C           - Calculate w
3006 C ---------------------------------------------------------------------------
3007                do i=1, 3
3008                  do j=1, 3
3009                    w_old(i,j) = Delta(i,j) * ( Lambda(i) * Eta(i) *
3010      +                          DXY((Kappa_old/Lambda(i)),Xi(i)) + V(i) )
3011                  enddo
3012                enddo
3013                CALL DA_inv_3(w_old,w_inv)
3014 C ---------------------------------------------------------------------------
3015 C           - Calculate M, dM_dPhi, ddM_dPHI
3016 C ---------------------------------------------------------------------------
3017                CALL M_IJKL_4_81(Phi_old,1,1,1,M,DM,D2M)
3018 C ---------------------------------------------------------------------------
```

```
C          - Calculate the new Y_ij
C -------------------------------------------------------------------
             CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C
             F_pot_new = F_pot(w_inv,Y_new,Gamma_old)
             g_new     = DXY(F_pot_new,F_power) - 1.d0
C -------------------------------------------------------------------
C          - Store the calculated values for each constituent
C ===================================================================
             IF (ID.eq.1) THEN
C ===================================================================
               CALL DTRANS_A_B_2(w_old,w_old_m,3)
               CALL DTRANS_A_B_2(w_inv,w_inv_m,3)
               CALL DTRANS_A_B_4(M,M_m,3)
               CALL DTRANS_A_B_6(DM,DM_m,3)
               CALL DTRANS_A_B_1(D2M,D2M_m,6561)
               CALL DTRANS_A_B_2(Y_new,Y_new_m,3)
               CALL DTRANS_A_B_2(Sigma_new,Sigma_new_m,3)
               CALL DTRANS_A_B_1(Xi,Xi_m,3)
               F_pot_new_m = F_pot_new
               g_new_m     = g_new
C ===================================================================
             ELSEIF (ID.eq.2) THEN
C ===================================================================
               CALL DTRANS_A_B_2(w_old,w_old_f,3)
               CALL DTRANS_A_B_2(w_inv,w_inv_f,3)
               CALL DTRANS_A_B_4(M,M_f,3)
               CALL DTRANS_A_B_6(DM,DM_f,3)
               CALL DTRANS_A_B_1(D2M,D2M_f,6561)
               CALL DTRANS_A_B_2(Y_new,Y_new_f,3)
               CALL DTRANS_A_B_2(Sigma_new,Sigma_new_f,3)
               CALL DTRANS_A_B_1(Xi,Xi_f,3)
               F_pot_new_f = F_pot_new
               g_new_f     = g_new
C ===================================================================
             ELSEIF (ID.eq.3) THEN
C ===================================================================
               CALL DTRANS_A_B_2(w_old,w_old_i,3)
               CALL DTRANS_A_B_2(w_inv,w_inv_i,3)
               CALL DTRANS_A_B_4(M,M_i,3)
               CALL DTRANS_A_B_6(DM,DM_i,3)
               CALL DTRANS_A_B_1(D2M,D2M_i,6561)
               CALL DTRANS_A_B_2(Y_new,Y_new_i,3)
               CALL DTRANS_A_B_2(Sigma_new,Sigma_new_i,3)
               CALL DTRANS_A_B_1(Xi,Xi_i,3)
               F_pot_new_i = F_pot_new
               g_new_i     = g_new
C ===================================================================
             ENDIF
C -------------------------------------------------------------------
C
C        --- Check if all conditions are satisfied
C
             IF (IFlag_damage.eq.0) THEN
C
C            --- There was no damage in the previous increment, hence
C                stepsize adjustment can be made. If there was damage in
C                the previous increment the stepsize is fixed for the
C                current increment to the fixed stepsize in case of
C                damage. This stepsize is fixed in the variable
C                dSigma_damage
C
               IF (IFlag_min_step.eq.0) THEN
C
C              --- minimum stepsize is lower then current stepsize
C                  ==> stepsize may be adjusted
C
                 IF (g_new.gt.0.d0.and.g_old.lt.0.d0) THEN
C
                   IFlag_stepsize(ID) = 0
C
                 ELSEIF (g_new.lt.0.d0.and.g_old.lt.0.d0) THEN
C
                   IFlag_stepsize(ID) = 1
C
                 ENDIF
C
               ELSEIF (IFlag_min_step.eq.1) THEN
C
C              --- minimum stepsize has been used in this increment
C
                 IFlag_stepsize(ID) = 1
C
               ENDIF
```

```
C
                IFlag_step = IFlag_step + IFlag_stepsize(ID)
C
              ELSEIF (IFlag_damage.eq.1) THEN
C
                IFlag_step = Number_of_Constituents
C
              ELSE
C
                WRITE(*,'(2(A/))')
     +            'An error occurred in the loading',
     +            'The variable IFlag_damage has a value > 1!!!'
                STOP
C
              ENDIF
C ---------------------------------------------------------------------
C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
2210        ENDDO
C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
C ---------------------------------------------------------------------
C          --- If conditions are not met entirely take appropriate actions
C
            IF (IFlag_step.lt.Number_of_Constituents) THEN
C
              STEP_factor      = 0.5d0
              Reduction_factor = Reduction_factor * STEP_factor
C
            ENDIF
C ---------------------------------------------------------------------
C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
2200      ENDDO
C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
C
C     --- Since all the conditions are satisfied for the current increment
C         calculate/update/reset a few necessary variables:
C         * calculate the new total stress
C         * Theta_old is replaced by Theta_new
C         * the flag IFlag_damage is reset to zero (0) for the current
C           increment and if damage occurs it will be set equal to 1
C
          CALL DA_ij_P_B_ij(Sigma_tot,dSigma_tot,Sigma_tot,3)
          Theta_old    = Theta_new
          IFlag_damage = 0
C ---------------------------------------------------------------------
C
C     --- Loop over the all the constituents to check for damage
C         with damage calculation if the criterion is satisfied
C
C ---------------------------------------------------------------------
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          DO 2300 ID = 1, Number_of_Constituents
C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
C
            CALL DINITIALIZE_ZERO_2(dPhi,3)
            IFlag_imp = 0
            IMP       = 'NN'
C ---------------------------------------------------------------------
C       --- Restore the variables as well as the previously calculated
C           values for each constituent
C
C ***********************************************************************
            IF (ID.eq.1) THEN
C ***********************************************************************
C             --- Matrix material ----------------------------
C
C             --- Constants
C
              CALL DTRANS_A_B_4(E_m_bar,E_bar,3)
              CALL DTRANS_A_B_4(E_m_bar_inv,E_bar_inv,3)
              CALL DTRANS_A_B_1(Eta_m,Eta,3)
              CALL DTRANS_A_B_1(Lambda_m,Lambda,3)
              do i=1,3
                V(i)  = Vi_m(i)
              enddo
              CALL DTRANS_A_B_1(Xi_m,Xi,3)
              c_Gamma   = c_Gamma_m
              F_power   = F_power_m
C
C             --- Variables which change in the process of computation
C
              CALL DTRANS_A_B_2(Sigma_m,Sigma_old,3)
              CALL DTRANS_A_B_2(dSigma_m,dSigma,3)
              CALL DTRANS_A_B_2(Phi_m,Phi_old,3)
```

```
3187              CALL DTRANS_A_B_2(Gamma_m,Gamma_old,3)
3188              CALL DTRANS_A_B_2(Y_m,Y_old,3)
3189              Kappa_old = Kappa_m
3190              g_old     = g_m
3191              F_pot_old = F_pot_m
3192 C
3193 C             --- Temporary intermediate results
3194 C
3195              CALL DTRANS_A_B_2(w_old_m,w_old,3)
3196              CALL DTRANS_A_B_2(w_inv_m,w_inv,3)
3197              CALL DTRANS_A_B_4(M_m,M,3)
3198              CALL DTRANS_A_B_6(DM_m,DM,3)
3199              CALL DTRANS_A_B_1(D2M_m,D2M,6561)
3200              CALL DTRANS_A_B_2(Y_new_m,Y_new,3)
3201              CALL DTRANS_A_B_2(Sigma_new_m,Sigma_new,3)
3202              F_pot_new = F_pot_new_m
3203              g_new     = g_new_m
3204 C
3205 C ------------------------------------------------------------------------
3206            ELSEIF (ID.eq.2) THEN
3207 C ------------------------------------------------------------------------
3208 C
3209 C             --- Fiber material ---------------------------------
3210 C
3211 C             --- Constants
3212 C
3213              CALL DTRANS_A_B_4(E_f_bar,E_bar,3)
3214              CALL DTRANS_A_B_4(E_f_bar_inv,E_bar_inv,3)
3215              CALL DTRANS_A_B_1(Eta_f,Eta,3)
3216              CALL DTRANS_A_B_1(Lambda_f,Lambda,3)
3217              do i=1,3
3218                V(i)  = Vi_f(i)
3219              enddo
3220              CALL DTRANS_A_B_1(Xi_f,Xi,3)
3221              c_Gamma   = c_Gamma_f
3222              F_power   = F_power_f
3223 C
3224 C             --- Variables which change in the process of computation
3225 C
3226              CALL DTRANS_A_B_2(Sigma_f,Sigma_old,3)
3227              CALL DTRANS_A_B_2(dSigma_f,dSigma,3)
3228              CALL DTRANS_A_B_2(Phi_f,Phi_old,3)
3229              CALL DTRANS_A_B_2(Gamma_f,Gamma_old,3)
3230              CALL DTRANS_A_B_2(Y_f,Y_old,3)
3231              Kappa_old = Kappa_f
3232              g_old     = g_f
3233              F_pot_old = F_pot_f
3234 C
3235 C             --- Temporary intermediate results
3236 C
3237              CALL DTRANS_A_B_2(w_old_f,w_old,3)
3238              CALL DTRANS_A_B_2(w_inv_f,w_inv,3)
3239              CALL DTRANS_A_B_4(M_f,M,3)
3240              CALL DTRANS_A_B_6(DM_f,DM,3)
3241              CALL DTRANS_A_B_1(D2M_f,D2M,6561)
3242              CALL DTRANS_A_B_2(Y_new_f,Y_new,3)
3243              CALL DTRANS_A_B_2(Sigma_new_f,Sigma_new,3)
3244              F_pot_new = F_pot_new_f
3245              g_new     = g_new_f
3246 C
3247 C ------------------------------------------------------------------------
3248            ELSEIF (ID.eq.3) THEN
3249 C ------------------------------------------------------------------------
3250 C
3251 C             --- Interface --------------------------------------
3252 C
3253 C             --- Constants
3254 C
3255              CALL DTRANS_A_B_4(E_i_bar,E_bar,3)
3256              CALL DTRANS_A_B_4(E_i_bar_inv,E_bar_inv,3)
3257              CALL DTRANS_A_B_1(Eta_i,Eta,3)
3258              CALL DTRANS_A_B_1(Lambda_i,Lambda,3)
3259              do i=1,3
3260                V(i)  = Vi_i(i)
3261              enddo
3262              CALL DTRANS_A_B_1(Xi_i,Xi,3)
3263              c_Gamma   = c_Gamma_i
3264              F_power   = F_power_i
3265 C
3266 C             --- Variables which change in the process of computation
3267 C
3268              CALL DTRANS_A_B_2(Sigma_i,Sigma_old,3)
3269              CALL DTRANS_A_B_2(dSigma_i,dSigma,3)
3270              CALL DTRANS_A_B_2(Phi_i,Phi_old,3)
```

```
3271              CALL DTRANS_A_B_2(Gamma_i,Gamma_old,3)
3272              CALL DTRANS_A_B_2(Y_i,Y_old,3)
3273              Kappa_old = Kappa_i
3274              g_old     = g_i
3275              F_pot_old = F_pot_i
3276 C
3277 C             --- Temporary intermediate results
3278 C
3279              CALL DTRANS_A_B_2(w_old_i,w_old,3)
3280              CALL DTRANS_A_B_2(w_inv_i,w_inv,3)
3281              CALL DTRANS_A_B_4(M_i,M,3)
3282              CALL DTRANS_A_B_6(DM_i,DM,3)
3283              CALL DTRANS_A_B_1(D2M_i,D2M,6561)
3284              CALL DTRANS_A_B_2(Y_new_i,Y_new,3)
3285              CALL DTRANS_A_B_2(Sigma_new_i,Sigma_new,3)
3286              F_pot_new = F_pot_new_i
3287              g_new     = g_new_i
3288 C ***************************************************************************
3289             ENDIF
3290 C ***************************************************************************
3291             CALL DTRANS_A_B_2(dSigma,dSigma_iter,3)
3292             IMP = 'NN'
3293             IFlag_imp = 0
3294 C -------------------------------------------------------------------------
3295 C
3296 C            Check if new state of stress will produce a damage state
3297 C
3298 C ###########################################################################
3299 C
3300             IF (g_old.lt.0.d0.and.g_new.ge.0.d0.and.
3301      +              dSigma(1,1).gt.0.d0) THEN
3302 C
3303 C ###########################################################################
3304               IMP = '1N'
3305               IFlag_damage_constituent(ID) = 1
3306 C -------------------------------------------------------------------------
3307 C             Material enters a damage state coming from a non-damaging
3308 C             state
3309 C
3310 C             ---> adjust stress increment such that a stress subincrement
3311 C                  is obtained which will bring the current subincrement
3312 C                  to the damage surface
3313 C
3314 C -------------------------------------------------------------------------
3315 C         --- Improvement according to W. F. Chen - Plasticity
3316 C -------------------------------------------------------------------------
3317               exponent  = F_power -1.d0
3318               dg_dF_pot = F_power * DXY(F_pot_old,exponent)
3319 C -------------------------------------------------------------------------
3320 C         --- Calculate dF_pot_dY_pq
3321 C -------------------------------------------------------------------------
3322               CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
3323 C -------------------------------------------------------------------------
3324 C         --- Calculate dY_ij_dSigma_mn
3325 C -------------------------------------------------------------------------
3326               CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
3327 C -------------------------------------------------------------------------
3328 C         --- Calculate dg_dSigma_ij
3329 C -------------------------------------------------------------------------
3330               CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
3331               CALL Dc_A_ij(dg_dF_pot,dg_dSigma,3)
3332 C -------------------------------------------------------------------------
3333 C         --- Calculate the parameter alpha
3334 C -------------------------------------------------------------------------
3335               CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha,3)
3336 C -------------------------------------------------------------------------
3337               alpha = - g_old / alpha
3338 C -------------------------------------------------------------------------
3339 C         --- Calculate the new stress based on parameter alpha
3340 C -------------------------------------------------------------------------
3341               CALL DTRANS_A_B_2(dSigma,dum2_1,3)
3342               CALL Dc_A_ij(alpha,dum2_1,3)
3343               CALL DA_ij_P_B_ij(Sigma_old,dum2_1,Sigma_new,3)
3344 C -------------------------------------------------------------------------
3345 C         --- Check damage surface with new stress
3346 C -------------------------------------------------------------------------
3347               CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
3348               F_pot_new = F_pot(w_inv,Y_new,Gamma_old)
3349               g_new     = DXY(F_pot_new,F_power) - 1.d0
3350 C -------------------------------------------------------------------------
3351 C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
3352 C
3353 C          R E M A I N D E R   O F   S T R E S S   I N C R E M E N T
3354 C
```

```
      C >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      C
      C  ---   Now calculate the damage increment for the remainder of
      C        the stress increment
      C ---------------------------------------------------------------------
                CALL DTRANS_A_B_2(Sigma_new,Sigma_nold,3)
                F_pot_old = F_pot_new
      C
                CALL DTRANS_A_B_2(dSigma_iter,dSigma,3)
      C
                const = 1.d0 - alpha
      C
                CALL Dc_A_ij(const,dSigma,3)
                CALL DTRANS_A_B_2(Y_new,Y_old,3)
      C ---------------------------------------------------------------------
      C        --- Calculate dg_dF_pot on the damage surface
      C ---------------------------------------------------------------------
                exponent  = F_power -1.d0
                dg_dF_pot = F_power * DXY(F_pot_old,exponent)
      C ---------------------------------------------------------------------
      C        --- Calculate dF_pot_dY_pq on the damage surface
      C ---------------------------------------------------------------------
                CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
      C ---------------------------------------------------------------------
      C        --- Calculate dY_ij_dSigma_mn on the damage surface
      C ---------------------------------------------------------------------
                CALL DYDSIGMA(dY_dSigma,Sigma_nold,M,DM,E_bar_inv)
      C ---------------------------------------------------------------------
      C        --- Calculate dY_ij_dPhi_mn on the damage surface
      C ---------------------------------------------------------------------
                CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_nold,M,DM,D2M)
      C ---------------------------------------------------------------------
      C        --- Calculate dF_pot_dGamma on the damage surface
      C ---------------------------------------------------------------------
                CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
      C ---------------------------------------------------------------------
      C        --- Calculate dGamma_dPhi on the damage surface
      C ---------------------------------------------------------------------
                CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
      C ---------------------------------------------------------------------
      C        --- Calculate dw_dKappa on the damage surface
      C ---------------------------------------------------------------------
                CALL DINITIALIZE_ZERO_2(dw_dKappa,3)
                do i=1, 3
                  exponent = Xi(i) - 1.d0
                  dw_dKappa(i,i) = Xi(i) * Eta(i)
     +                     * DXY((Kappa_old/Lambda(i)),exponent)
                enddo
      C ---------------------------------------------------------------------
      C        --- Calculate dF_pot_dw on the damage surface
      C ---------------------------------------------------------------------
                CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
      C ---------------------------------------------------------------------
      C        --- Calculate the new stress level
      C ---------------------------------------------------------------------
                CALL DA_ij_P_B_ij(Sigma_old,dSigma_iter,Sigma_new,3)
      C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      C  ---    Calculate the derivatives of g wrt Phi   ---> dg_dPhi
      C                                      g wrt Y     ---> dg_dY
      C                                      g wrt Gamma ---> dg_dGamma
      C                                      g wrt Sigma ---> dg_dSigma
      C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      C ---------------------------------------------------------------------
      C        --- Calculate dg_dSigma
      C ---------------------------------------------------------------------
                CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
                CALL DC_A_ij(dg_dF_pot,dg_dSigma,3)
      C ---------------------------------------------------------------------
      C        --- Calculate dg_dY on the damage surface
      C ---------------------------------------------------------------------
                CALL DTRANS_A_B_2(dF_pot_dY,dg_dY,3)
      C
                CALL Dc_A_ij(dg_dF_pot,dg_dY,3)
      C ---------------------------------------------------------------------
      C        --- Calculate dg_dKappa on the damage surface
      C ---------------------------------------------------------------------
                CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
                dg_dKappa = dg_dF_pot * dg_dKappa
      C ---------------------------------------------------------------------
      C        --- Calculate dg_dPhi on the damage surface
      C ---------------------------------------------------------------------
                CALL DINITIALIZE_ZERO_2(dum2_1,3)
                CALL DINITIALIZE_ZERO_2(dum2_2,3)
      C        --- 1. part
```

```
3439              CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
3440 C         --- 2. part
3441              CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
3442 C         --- add the 2 parts
3443              CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
3444              CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
3445 C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3446 C
3447 C         --- Calculate Psi as
3448 C
3449 C                   Psi = - Psi_num / Psi_den
3450 C
3451 C         where:    Psi_num_ijkl = dg_dY_ij dg_dSigma_kl
3452 C
3453 C                   Psi_den      = ( dg_dPhi_mn + dg_dKappa * Y_mn
3454 C                                    - c * dg_dY_mn ) * dg_dY_mn
3455 C
3456 C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3457 C ---------------------------------------------------------------------
3458 C         --- Calculate Psi_den
3459 C ---------------------------------------------------------------------
3460              CALL DTRANS_A_B_2(dg_dY,dum2_1,3)
3461              CALL Dc_A_ij(c_Gamma,dum2_1,3)
3462              CALL DTRANS_A_B_2(Y_new,dum2_2,3)
3463              CALL Dc_A_ij(dg_dKappa,dum2_2,3)
3464              CALL DA_ij_M_B_ij(dum2_2,dum2_1,dum2_3,3)
3465              CALL DA_ij_P_B_ij(dum2_3,dg_dPhi,dum2_2,3)
3466              CALL DA_ij_B_ij(dum2_2,dg_dY,Psi_den,3)
3467 C ---------------------------------------------------------------------
3468 C         --- Calculate Psi_num
3469 C ---------------------------------------------------------------------
3470              CALL DA_ij_B_kl(dg_dY,dg_dSigma,Psi_num,3)
3471 C ---------------------------------------------------------------------
3472 C         --- Calculate Psi
3473 C ---------------------------------------------------------------------
3474              CALL DTRANS_A_B_4(Psi_num,Psi,3)
3475              Psi_den_inv = -1.d0 / Psi_den
3476              CALL Dc_A_ijkl(Psi_den_inv,Psi,3)
3477 C ---------------------------------------------------------------------
3478 C         --- Calculate the damage increment dPhi
3479 C ---------------------------------------------------------------------
3480              CALL DA_ijkl_B_kl(Psi,dSigma,dPhi,3)
3481 C ---------------------------------------------------------------------
3482 C    ---   Update the damage variables
3483 C ---------------------------------------------------------------------
3484 C
3485 C         --- Store the old value of Phi
3486 C
3487              CALL DTRANS_A_B_2(Phi_old,Phi_dum,3)
3488 C
3489 C         --- Update Phi
3490 C
3491              CALL DA_ij_P_B_ij(Phi_old,dPhi,Phi_new,3)
3492 C
3493 C         --- Update M, DM, DDM
3494 C
3495              CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
3496 C
3497 C         --- Update Y
3498 C
3499              CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
3500 C
3501 C         --- Update Kappa
3502 C
3503              CALL DA_ij_B_ij(Y_old,dPhi,dKappa,3)
3504              Kappa_new = Kappa_old + dKappa
3505 C
3506 C         --- Calculate w_new and w_inv
3507 C
3508              do i=1, 3
3509                do j=1, 3
3510                  w_new(i,j) = Delta(i,j) * (Lambda(i) * Eta(i) *
3511     +                         DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i))
3512                enddo
3513              enddo
3514 C
3515              CALL DA_inv_3(w_new,w_inv)
3516 C
3517 C         --- Update Gamma
3518 C
3519              CALL DTRANS_A_B_2(dPhi,dum2_1,3)
3520              CALL Dc_A_ij(c_Gamma,dum2_1,3)
3521              CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
3522 C
```

```
C        --- Update F_pot
C
            F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C
C        --- Calculate dY
C
            CALL DA_ij_M_B_ij(Y_new,Y_old,dY,3)
C
C        --- Calculate new g
C
            g_new = DXY(F_pot_new,F_power) - 1.d0
C
C        --- Check for g
C
C#########################################################################
            IF (dabs(g_new).gt.1.d-4) THEN
C#########################################################################
C
C         ---> Adjust damage variables Phi, Kappa and Gamma such that
C              the damage surface is satisfied using a Taylor series
C              expansion of order 1 based on the following relationships:
C
C             dKappa = Y dPhi
C             dGamma = c dPhi
C
              IMP = '1Y'
C-------------------------------------------------------------------------
C          --- Calculate dg_dF_pot
C-------------------------------------------------------------------------
              exponent  = F_power - 1.d0
              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
C-------------------------------------------------------------------------
C          --- Calculate dF_pot_dY_pq
C-------------------------------------------------------------------------
              CALL DA_inv_3(w_old,w_inv)
              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
C-------------------------------------------------------------------------
C          --- Calculate M, DM, DDM based on the variables at the end of
C              the previous step = damage surface is satisfied
C-------------------------------------------------------------------------
              CALL M_IJKL_4_81(Phi_old,1,1,1,M,DM,D2M)
C-------------------------------------------------------------------------
C          --- Calculate g at the end of the previous step
C-------------------------------------------------------------------------
              g_old = DXY(F_pot_old,F_power) - 1.d0
C-------------------------------------------------------------------------
C          --- Calculate dg_dSigma at the end of the previous step
C-------------------------------------------------------------------------
              CALL DYDSIGMA(dY_dSigma,Sigma_nold,M,DM,E_bar_inv)
              CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dF_pot_dSigma,3)
              CALL Dc_A_ij(dg_dF_pot,dF_pot_dSigma,3)
              CALL DTRANS_A_B_2(dF_pot_dSigma,dg_dSigma,3)
C-------------------------------------------------------------------------
C          --- Calculate dg_dKappa at the end of the previous step
C-------------------------------------------------------------------------
              CALL DWDKAPPA(dw_dKappa,Xi,Eta,Lambda,Kappa_old)
              CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
              CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
              dg_dKappa = dg_dF_pot * dg_dKappa
C-------------------------------------------------------------------------
C          --- Calculate dg_dPhi at the end of the previous step
C-------------------------------------------------------------------------
              CALL DINITIALIZE_ZERO_2(dum2_1,3)
              CALL DINITIALIZE_ZERO_2(dum2_2,3)
              CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
              CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
              CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_nold,M,DM,D2M)
C          --- 1. part
              CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C          --- 2. part
              CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
C          --- add the 2 parts
              CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C-------------------------------------------------------------------------
C          --- Calculate dg_dGamma at the end of the previous step
C              Note: dg_dGamma = - dg_dY
C-------------------------------------------------------------------------
              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
              const = -1.d0
              CALL Dc_A_ij(const,dF_pot_dY,3)
              CALL DTRANS_A_B_2(dF_pot_dY,dg_dGamma,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dGamma,3)
C-------------------------------------------------------------------------
```

```
C          --- Calculate: alpha = - alpha_num / alpha_den
C -----------------------------------------------------------------------
               CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha_num,3)
               alpha_num = alpha_num + g_old
C
               CALL DTRANS_A_B_2(Y_old,dum2_1,3)
               CALL Dc_A_ij(dg_dKappa,dum2_1,3)
               CALL DA_ij_P_B_ij(dg_dPhi,dum2_1,dum2_2,3)
               CALL DTRANS_A_B_2(dg_dGamma,dum2_1,3)
               CALL Dc_A_ij(c_Gamma,dum2_1,3)
               CALL DA_ij_P_B_ij(dum2_2,dum2_1,dum2_3,3)
               CALL DA_ij_B_ij(dum2_3,dPhi,alpha_den,3)
C
               alpha_s = - alpha_num / alpha_den
C -----------------------------------------------------------------------
C       --- Now calculate/update the damage affected variables Phi,
C          Kappa and Gamma, and then update the toher variables which
C          depend on them
C -----------------------------------------------------------------------
C
C          --- Update Kappa
C
               Kappa_new = Kappa_old + alpha_s * dKappa
C
C          --- Update Phi
C
               CALL DTRANS_A_B_2(dPhi,dum2_1,3)
               CALL Dc_A_ij(alpha_s,dum2_1,3)
               CALL DA_ij_P_B_ij(Phi_old,dum2_1,Phi_new,3)
C
C          --- Update Gamma
C
               CALL Dc_A_ij(c_Gamma,dum2_1,3)
               CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
C
C          --- Calculate w_new and w_inv
C
               do i=1, 3
                 do j=1, 3
                   w_new(i,j) = Delta(i,j) * ( Lambda(i) * Eta(i) *
     +                          DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i))
                 enddo
               enddo
               CALL DA_inv_3(w_new,w_inv)
C
C          --- Update M, DM, DDM
C
               CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
C
C          --- Update Y
C
               CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C
C          --- Update F_pot
C
               F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C
C          --- Calculate new g
C
               g_new = DXY(F_pot_new,F_power) - 1.d0
C -----------------------------------------------------------------------
             ENDIF
C -----------------------------------------------------------------------
C
C ######################################################################
C
           ELSEIF (g_old.gt.0.d0.and.g_new.gt.0.d0.and.
     +                       dSigma(1,1).gt.0.d0) THEN
C
C ######################################################################
C
             IMP = '2N'
             IFlag_damage_constituent(ID) = 1
C
             IF (ID.eq.1) I_count = I_count + 1
C
C ----------------------------------------------------------------------
C
C          C O M I N G   F R O M   A   D A M A G E   S T A T E
C
C ----------------------------------------------------------------------
C
             CALL DTRANS_A_B_2(dSigma_iter,dSigma,3)
C -----------------------------------------------------------------------
```

```
C         --- Calculation of dg_dY
C ------------------------------------------------------------------
C           - Calculate dg_dF_pot
C
              exponent  = F_power -1.d0
              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
C
C           - Calculate dF_pot_dY_pq
C
              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
C ------------------------------------------------------------------
C           - now calculate dg_dY
C
              CALL DTRANS_A_B_2(dF_pot_dY,dg_dY,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dY,3)
C ------------------------------------------------------------------
C         --- Calculate dY_ij_dSigma_mn
C ------------------------------------------------------------------
              CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
C ------------------------------------------------------------------
C         --- Calculate dY_ij_dPhi_mn
C ------------------------------------------------------------------
              CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_old,M,DM,D2M)
C ------------------------------------------------------------------
C         --- Calculate dF_pot_dGamma
C ------------------------------------------------------------------
              CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
C ------------------------------------------------------------------
C         --- Calculate dGamma_dPhi
C ------------------------------------------------------------------
              CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
C ------------------------------------------------------------------
C         --- Calculate dw_dKappa
C ------------------------------------------------------------------
              CALL DINITIALIZE_ZERO_2(dw_dKappa,3)
              do i=1, 3
                exponent = Xi(i) - 1.d0
                dw_dKappa(i,i) = Xi(i) * Eta(i)
     +                   * DXY((Kappa_old/Lambda(i)),exponent)
              enddo
              exponent = Xi(1) - 1.d0
C ------------------------------------------------------------------
C         --- Calculate dF_pot_dw
C ------------------------------------------------------------------
              CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
C ------------------------------------------------------------------
C         --- Calculate the new stress level
C ------------------------------------------------------------------
              CALL DA_ij_P_B_ij(Sigma_old,dSigma_iter,Sigma_new,3)
C ------------------------------------------------------------------
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C   ---   Calculate the derivatives of g wrt Phi   ---> dg_dPhi
C                                      g wrt Y     ---> dg_dY
C                                      g wrt Gamma ---> dg_dGamma
C                                      g wrt SIgma ---> dg_dSigma
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ------------------------------------------------------------------
C         --- Calculate dg_dSigma
C ------------------------------------------------------------------
              CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dg_dSigma,3)
              CALL DC_A_ij(dg_dF_pot,dg_dSigma,3)
C ------------------------------------------------------------------
C         --- Calculate dg_dY
C ------------------------------------------------------------------
              CALL Dc_A_ij(dg_dF_pot,dF_pot_dY,3)
C ------------------------------------------------------------------
C         --- Calculate dg_dKappa
C ------------------------------------------------------------------
              CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
              dg_dKappa = dg_dF_pot * dg_dKappa
C ------------------------------------------------------------------
C         --- Calculate dg_dPhi on the damage surface
C ------------------------------------------------------------------
              CALL DINITIALIZE_ZERO_2(dum2_1,3)
              CALL DINITIALIZE_ZERO_2(dum2_2,3)
C         --- 1. part
              CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C         --- 2. part
              CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
C         --- add the 2 parts
              CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
```

```
C     -   Calculate Psi as
C
C                 Psi = - Psi_num / Psi_den
C
C         where:    Psi_num_ijkl = dg_dY_ij dg_dSigma_kl
C
C                 Psi_den      = ( dg_dPhi_mn + dg_dKappa * Y_mn
C                                  - c * dg_dY_mn ) * dg_dY_mn
C
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ----------------------------------------------------------------------
C         --- Calculate Psi_den
C ----------------------------------------------------------------------
              CALL DTRANS_A_B_2(dg_dY,dum2_1,3)
              CALL Dc_A_ij(c_Gamma,dum2_1,3)
              CALL DTRANS_A_B_2(Y_new,dum2_2,3)
              CALL Dc_A_ij(dg_dKappa,dum2_2,3)
              CALL DA_ij_M_B_ij(dum2_2,dum2_1,dum2_3,3)
              CALL DA_ij_P_B_ij(dum2_3,dg_dPhi,dum2_2,3)
              CALL DA_ij_B_ij(dum2_2,dg_dY,Psi_den,3)
C ----------------------------------------------------------------------
C         --- Calculate Psi_num
C ----------------------------------------------------------------------
              CALL DA_ij_B_kl(dg_dY,dg_dSigma,Psi_num,3)
C ----------------------------------------------------------------------
C         --- Calculate Psi
C ----------------------------------------------------------------------
              CALL DTRANS_A_B_4(Psi_num,Psi,3)
              Psi_den_inv = -1.d0 / Psi_den
              CALL Dc_A_ijkl(Psi_den_inv,Psi,3)
C ----------------------------------------------------------------------
C         --- Calculate the damage increment dPhi
C ----------------------------------------------------------------------
              CALL DA_ijkl_B_kl(Psi,dSigma,dPhi,3)
C ----------------------------------------------------------------------
C     --- Now update the damage variables
C ----------------------------------------------------------------------
C
C         --- Store the old value of Phi
C
              CALL DTRANS_A_B_2(Phi_old,Phi_dum,3)
C
C         --- Update Phi
C
              CALL DA_ij_P_B_ij(Phi_old,dPhi,Phi_new,3)
C
C         --- Update M, DM, DDM
C
              CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
C
C         --- Update Y
C
              CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
C
C         --- Update Kappa
C
              CALL DA_ij_B_ij(Y_old,dPhi,dKappa,3)
              Kappa_new = Kappa_old + dKappa
C
C         --- Calculate w_new and w_inv
C
              do i=1, 3
                do j=1, 3
                  w_new(i,j) = Delta(i,j) * (Lambda(i) * Eta(i) *
     +                         DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i))
                enddo
              enddo
              CALL DA_inv_3(w_new,w_inv)
C
C         --- Update Gamma
C
              CALL DTRANS_A_B_2(dPhi,dum2_1,3)
              CALL Dc_A_ij(c_Gamma,dum2_1,3)
              CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
C
C         --- Update F_pot
C
              F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
C
C         --- Calculate dY
C
              CALL DA_ij_M_B_ij(Y_new,Y_old,dY,3)
C
C         --- Calculate new g
```

```
C
            g_new = DXY(F_pot_new,F_power) - 1.d0
C
C           --- Check for g
C
C ######################################################################
            IF (DABS(g_new).gt.1.d-4) THEN
C ######################################################################
C
C         ---> Adjust damage variables Phi, Kappa and Gamma such that
C              the damage surface is satisfied using a Taylor series
C              expansion of order 1 based on the following relationships:
C
C             dKappa = Y dPhi
C             dGamma = c dPhi
C
              IMP = '2Y'
C ----------------------------------------------------------------------
C         --- Calculate dg_dF_pot
C ----------------------------------------------------------------------
              exponent  = F_power -1.d0
              dg_dF_pot = F_power * DXY(F_pot_old,exponent)
C ----------------------------------------------------------------------
C         --- Calculate dF_pot_dY_pq
C ----------------------------------------------------------------------
              CALL DA_inv_3(w_old,w_inv)
              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
C ----------------------------------------------------------------------
C         --- Calculate M, DM, DDM based on the variables at the end of
C             the previous step = damage surface is satisfied
C ----------------------------------------------------------------------
              CALL M_IJKL_4_81(Phi_old,1,1,1,M,DM,D2M)
               CALL DA_inv_3(w_old,w_inv)
C ----------------------------------------------------------------------
C         --- Calculate g at the end of the previous step
C ----------------------------------------------------------------------
              g_old = DXY(F_pot_old,F_power) - 1.d0
C ----------------------------------------------------------------------
C         --- Calculate dg_dSigma at the end of the previous step
C             Make sure that all the variables used are evaluated at
C             the same increment level
C ----------------------------------------------------------------------
              CALL DYDSIGMA(dY_dSigma,Sigma_old,M,DM,E_bar_inv)
              CALL DA_ijkl_B_ij(dY_dSigma,dF_pot_dY,dF_pot_dSigma,3)
              CALL Dc_A_ij(dg_dF_pot,dF_pot_dSigma,3)
              CALL DTRANS_A_B_2(dF_pot_dSigma,dg_dSigma,3)
C ----------------------------------------------------------------------
C         --- Calculate dg_dKappa at the end of the previous step
C ----------------------------------------------------------------------
              CALL DWDKAPPA(dw_dKappa,Xi,Eta,Lambda,Kappa_old)
              CALL DFDW(dF_pot_dw,w_inv,Y_old,Gamma_old)
              CALL DA_ij_B_ij(dF_pot_dw,dw_dKappa,dg_dKappa,3)
              dg_dKappa = dg_dF_pot * dg_dKappa
C ----------------------------------------------------------------------
C         --- Calculate dg_dPhi at the end of the previous step
C ----------------------------------------------------------------------
              CALL DINITIALIZE_ZERO_2(dum2_1,3)
              CALL DINITIALIZE_ZERO_2(dum2_2,3)
              CALL DGAMMADPHI(dGamma_dPhi,Delta,c_Gamma)
              CALL DFDGAMMA(dF_pot_dGamma,w_inv,Y_old,Gamma_old)
              CALL DYDPHI_81(dY_dPhi,E_bar_inv,Sigma_old,M,DM,D2M)
C         --- 1. part
              CALL DA_ijkl_B_ij(dY_dPhi,dF_pot_dY,dum2_1,3)
C         --- 2. part
              CALL DA_ijkl_B_ij(dGamma_dPhi,dF_pot_dGamma,dum2_2,3)
C         --- add the 2 parts
              CALL DA_ij_P_B_ij(dum2_1,dum2_2,dg_dPhi,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dPhi,3)
C ----------------------------------------------------------------------
C         --- Calculate dg_dGamma at the end of the previous step
C             Note: dg_dGamma = - dg_dY
C ----------------------------------------------------------------------
              CALL DFDY(dF_pot_dY,w_inv,Y_old,Gamma_old)
              const = -1.d0
              CALL Dc_A_ij(const,dF_pot_dY,3)
              CALL DTRANS_A_B_2(dF_pot_dY,dg_dGamma,3)
              CALL Dc_A_ij(dg_dF_pot,dg_dGamma,3)
C ----------------------------------------------------------------------
C         --- Calculate alpha = - alpha_num / alpha_den
C ----------------------------------------------------------------------
              CALL DA_ij_B_ij(dg_dSigma,dSigma,alpha_num,3)
              alpha_num = alpha_num + g_old
C
              CALL DTRANS_A_B_2(Y_old,dum2_1,3)
```

```
3943                CALL Dc_A_ij(dg_dKappa,dum2_1,3)
3944                CALL DA_ij_P_B_ij(dg_dPhi,dum2_1,dum2_2,3)
3945                CALL DTRANS_A_B_2(dg_dGamma,dum2_1,3)
3946                CALL Dc_A_ij(c_Gamma,dum2_1,3)
3947                CALL DA_ij_P_B_ij(dum2_2,dum2_1,dum2_3,3)
3948                CALL DA_ij_B_ij(dum2_3,dPhi,alpha_den,3)
3949 C
3950                alpha_s = - alpha_num / alpha_den
3951 C ------------------------------------------------------------------
3952 C           - Now calculate/update the damage affected variables Phi,
3953 C             Kappa and Gamma and then recalculate the quantities which
3954 C             depend on them.
3955 C ------------------------------------------------------------------
3956 C
3957 C         --- Update Phi
3958 C
3959                CALL DTRANS_A_B_2(dPhi,dum2_1,3)
3960                CALL Dc_A_ij(alpha_s,dum2_1,3)
3961                CALL DA_ij_P_B_ij(Phi_old,dum2_1,Phi_new,3)
3962 C
3963 C         --- Update Kappa
3964 C
3965                Kappa_new = Kappa_old + alpha_s * dKappa
3966 C
3967 C         --- Update Gamma
3968 C
3969                CALL Dc_A_ij(c_Gamma,dum2_1,3)
3970                CALL DA_ij_P_B_ij(Gamma_old,dum2_1,Gamma_new,3)
3971 C
3972 C         --- Calculate w_new and w_inv
3973 C
3974                do i=1, 3
3975                  do j=1, 3
3976                    w_new(i,j) = Delta(i,j) * ( Lambda(i) * Eta(i) *
3977      +                          DXY((Kappa_new/Lambda(i)),Xi(i)) + V(i))
3978                  enddo
3979                enddo
3980                CALL DA_inv_3(w_new,w_inv)
3981 C
3982 C         --- Update M, DM, DDM
3983 C
3984              CALL M_IJKL_4_81(Phi_new,1,1,1,M,DM,D2M)
3985 C
3986 C         --- Update Y
3987 C
3988                 CALL Y_ij(Y_new,E_bar_inv,M,DM,Sigma_new)
3989 C
3990 C         --- Update F_pot
3991 C
3992                   F_pot_new = F_pot(w_inv,Y_new,Gamma_new)
3993 C
3994 C         --- Calculate new g
3995 C
3996                   g_new = DXY(F_pot_new,F_power) - 1.d0
3997 C
3998 C ------------------------------------------------------------------
3999                ENDIF
4000 C
4001 C ------------------------------------------------------------------
4002 C
4003              ELSEIF ((g_old.gt.0.d0.and.g_new.le.0.d0).or.
4004      +               (g_old.le.0.d0.and.g_new.le.0.d0).or.
4005      +               (g_old.gt.0.d0.and.g_new.gt.0.d0.and.
4006      +                           dSigma(1,1).lt.0.d0).or.
4007      +               (g_old.lt.0.d0.and.g_new.ge.0.d0.and.
4008      +                           dSigma(1,1).lt.0.d0)
4009      +                                              ) THEN
4010 C
4011 C ------------------------------------------------------------------
4012 C
4013 C             No damage neither previous increment nor in the current
4014 C
4015 C ------------------------------------------------------------------
4016                IFlag_damage_constituent(ID) = 0
4017                CALL DTRANS_A_B_2(Phi_old,Phi_new,3)
4018                Kappa_new = Kappa_old
4019                CALL DTRANS_A_B_2(Gamma_old,Gamma_new,3)
4020 C ------------------------------------------------------------------
4021              ENDIF
4022 C
4023 C ------------------------------------------------------------------
4024 C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
4025 C
4026 C         Store Variables in the appropriate places for each constituent
```

```
C
CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C*****************************************************************
      IF (ID.eq.1) THEN
C*****************************************************************
C
C     ----- Matrix material ------
C
C-----------------------------------------------------------------
C     Check for inconsistency in form of healing in the constituents
C-----------------------------------------------------------------
C
        IF (Phi_new(1,1)-Phi_m(1,1).LT.-1.d-5)
     +    CALL Damage_healing(ID,L_inc,NOC,NOC_old,g_old,g_new,
     +    Gamma_old,Gamma_new,Y_new,Y_old,Kappa_new,
     +    Kappa_old,F_pot_new,F_pot_old,Phi_m,Phi_new,dPhi)
C
C-----------------------------------------------------------------
C
        CALL DTRANS_A_B_2(Sigma_new,Sigma_m,3)
        CALL DTRANS_A_B_2(Phi_new,Phi_m,3)
        CALL DTRANS_A_B_2(Gamma_new,Gamma_m,3)
        CALL DTRANS_A_B_2(Y_new,Y_m,3)
        Kappa_m = Kappa_new
        g_m     = g_new
        F_pot_m = F_pot_new
        If (Kappa_m_0.lt.1.d-19) Kappa_m_0 = Kappa_new
C-----------------------------------------------------------------
C     Printout to check
C
        IF (NOC.le.4)
     +  write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A2,1x,
     +                        d12.6,2x,f6.3,2x,f8.5,3x,f10.5)')
     +    L_inc,
     +    g_m,Y_m(1,1),F_pot_m,Phi_m(1,1),Xi(1),Sigma_m(1,1),IMP,
     +    Kappa_m,c_m_adj,w_old(1,1),Sigma_tot(1,1)
C-----------------------------------------------------------------
        IF (Phi_m(1,1).gt.Phi_m_cr(1,1)) IFlag_failed = 1
C*****************************************************************
      ELSEIF (ID.eq.2) THEN
C*****************************************************************
C
C     ------ Fiber material ------
C
C-----------------------------------------------------------------
C     Check for inconsistency in form of healing in the constituents
C-----------------------------------------------------------------
C
        IF (Phi_new(1,1)-Phi_f(1,1).LT.-1.d-5)
     +    CALL Damage_healing(ID,L_inc,NOC,NOC_old,g_old,g_new,
     +    Gamma_old,Gamma_new,Y_new,Y_old,Kappa_new,
     +    Kappa_old,F_pot_new,F_pot_old,Phi_f,Phi_new,dPhi)
C
C-----------------------------------------------------------------
C
        CALL DTRANS_A_B_2(Sigma_new,Sigma_f,3)
        CALL DTRANS_A_B_2(Phi_new,Phi_f,3)
        CALL DTRANS_A_B_2(Y_new,Y_f,3)
        CALL DTRANS_A_B_2(Gamma_new,Gamma_f,3)
        Kappa_f = Kappa_new
        g_f     = g_new
        F_pot_f = F_pot_new
        If (Kappa_f_0.lt.1.d-19) Kappa_f_0 = Kappa_new
C-----------------------------------------------------------------
C     Printout to check
C-----------------------------------------------------------------
        IF (NOC.le.4)
     +    write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A,1x,
     +                              d12.6,2x,f6.3,2x,f8.5)')
     +     L_inc,
     +     g_f,Y_f(1,1),F_pot_f,Phi_f(1,1),Xi(1),Sigma_f(1,1),IMP,
     +     Kappa_f,c_f_adj,w_old(1,1)
C-----------------------------------------------------------------
        IF (Phi_f(1,1).gt.Phi_f_cr(1,1)) IFlag_failed = 2
C*****************************************************************
      ELSEIF (ID.eq.3) THEN
C*****************************************************************
C
C     ------ Interface -----------
C
C-----------------------------------------------------------------
C     Check for inconsistency in form of healing in the constituents
C-----------------------------------------------------------------
C
```

```
4111              IF (Phi_new(1,1)-Phi_i(1,1).LT.-1.d-5)
4112     +          CALL Damage_healing(ID,L_inc,NOC,NOC_old,g_old,g_new,
4113     +          Gamma_old,Gamma_new,Y_new,Y_old,Kappa_new,
4114     +          Kappa_old,F_pot_new,F_pot_old,Phi_i,Phi_new,dPhi)
4115 C -------------------------------------------------------------------
4116               CALL DTRANS_A_B_2(Sigma_new,Sigma_i,3)
4117               CALL DTRANS_A_B_2(Phi_new,Phi_i,3)
4118               CALL DTRANS_A_B_2(Y_new,Y_i,3)
4119               CALL DTRANS_A_B_2(Gamma_new,Gamma_i,3)
4120               Kappa_i = Kappa_new
4121               g_i     = g_new
4122               F_pot_i = F_pot_new
4123               If (Kappa_i_0.lt.1.d-19) Kappa_i_0 = Kappa_new
4124 C -------------------------------------------------------------------
4125 C           Printout to check
4126 C -------------------------------------------------------------------
4127               IF (NOC.le.4)
4128     +          write(Check_c(ID),'(I12,5(2x,d14.8),2x,f9.3,1x,A,1x,
4129     +                                        d12.6,4x,A,3x,f8.5)')
4130     +           L_inc,
4131     +           g_i,Y_i(1,1),F_pot_i,Phi_i(1,1),Xi(1),Sigma_i(1,1),IMP,
4132     +           Kappa_i,'N/A',w_old(1,1)
4133 C -------------------------------------------------------------------
4134               IF (Phi_i(1,1).gt.Phi_i_cr(1,1)) IFlag_failed = 3
4135 C ===================================================================
4136             ENDIF
4137 C ===================================================================
4138 C
4139 C   Abort the run if one of the damage components is bigger than a
4140 C   critical value. Print the appropriate variables and stop.
4141 C
4142 C -------------------------------------------------------------------
4143             IF (IFlag_damage_constituent(ID).eq.1) Iflag_damage = 1
4144 C -------------------------------------------------------------------
4145 C
4146 C     --- End of DO Loop on the constituents for the current increment
4147 C
4148 C -------------------------------------------------------------------
4149 C
4150 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
4151 2300      ENDDD
4152 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
4153 C
4154 C -------------------------------------------------------------------
4155         IF (Number_Of_Cycles.le.1)
4156     +    CALL STRESS_STRAIN_DATA(N_inc_old+N_inc_mean+
4157     +                            (NOC-1)*4*N_inc+L_inc,
4158     +              Sigma_m,Sigma_f,
4159     +              E_m_bar_inv,E_f_bar_inv,M_m,M_f,
4160     +              Epsilon_m,Epsilon_f,SS_dat)
4161 C -------------------------------------------------------------------
4162 C
4163 C     --- End of DO Loop on the current loading cycle
4164 C
4165 C -------------------------------------------------------------------
4166 C
4167 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
4168 2100    ENDDO
4169 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
4170 C
4171 C -------------------------------------------------------------------
4172 C     --- Calculate deformation after each cycle
4173 C -------------------------------------------------------------------
4174         IF (Number_Of_Cycles.gt.1)
4175     +    CALL STRESS_STRAIN_DATA(NOC_old+NOC,Sigma_m,Sigma_f,
4176     +              E_m_bar_inv,E_f_bar_inv,M_m,M_f,
4177     +              Epsilon_m,Epsilon_f,SS_dat)
4178 C -------------------------------------------------------------------
4179 C   --- Calculate the overall composite damage combining the damages in
4180 C       the constituents
4181 C       However if interface damage has to be considered calculate first
4182 C       the damage effect tensor due to interface damage to be used in
4183 C       the calculation for the overall damage
4184 C -------------------------------------------------------------------
4185         CALL COMPOSITE_DAMAGE(c_f,c_m,Phi_f,Phi_m,Phi_c)
4186 C -------------------------------------------------------------------
4187 C
4188         IF (IFlag_write.eq.1) THEN
4189 C
4190           IF (IFlag_Interface.eq.1) THEN
4191             write(Phi_dat,'(1x,I6,11(1x,d14.8))')
4192     +          NOC_old+NOC,Phi_m(1,1),Sigma_m(1,1),Xi_m(1),
4193     +          Phi_f(1,1),Sigma_f(1,1),Xi_f(1),Phi_i(1,1),Sigma_i(1,1),
4194     +          Xi_i(1),Phi_c(1,1),Sigma_tot(1,1)
```

```
4195          ELSEIF (IFlag_Interface.eq.0) THEN
4196            write(Phi_dat,'(1x,I6,8(3x,d14.8))')
4197     +         NOC_old+NOC,Phi_m(1,1),Sigma_m(1,1),Xi_m(1),
4198     +         Phi_f(1,1),Sigma_f(1,1),Xi_f(1),Phi_c(1,1),Sigma_tot(1,1)
4199          ENDIF
4200 C
4201        ENDIF
4202 C -------------------------------------------------------------------
4203 C
4204 C     Store the results in an array which is printed at the end of every
4205 C     1000 cycles
4206 C
4207 C -------------------------------------------------------------------
4208        Results_damage(NOC_old+NOC,1) = Phi_m(1,1)
4209        Results_damage(NOC_old+NOC,2) = Sigma_m(1,1)
4210        Results_damage(NOC_old+NOC,3) = Xi_m(1)
4211        Results_damage(NOC_old+NOC,4) = Phi_f(1,1)
4212        Results_damage(NOC_old+NOC,5) = Sigma_f(1,1)
4213        Results_damage(NOC_old+NOC,6) = Xi_f(1)
4214        Results_damage(NOC_old+NOC,7) = Phi_c(1,1)
4215        Results_damage(NOC_old+NOC,8) = Sigma_tot(1,1)
4216 C -------------------------------------------------------------------
4217        EVF(NOC_old+NOC,1) = c_m_adj
4218        EVF(NOC_old+NOC,2) = c_f_adj
4219 C
4220 C -------------------------------------------------------------------
4221 C
4222 C ------ Material failure !!!!!!!! ----------------------------------
4223 C
4224 C -------------------------------------------------------------------
4225        IF (IFlag_failed.eq.1.or.IFlag_failed.eq.2) THEN
4226 C
4227 C        Write results in array to file
4228 C
4229          IF (IFlag_write.eq.0) THEN
4230 C
4231            DO i=1, NOC
4232 C
4233              write(Phi_dat,'(1x,I6,8(3x,d14.8))')
4234     +              NOC_old+NOC,(Results_damage(i,j),j=1,8)
4235 C
4236            ENDDO
4237 C
4238            DO i=1, NOC
4239 C
4240              write(Check_r,'(1x,I6,2(3x,f12.6))')
4241     +              NOC_old+1,(EVF(i,j),j=1,2)
4242 C
4243            ENDDO
4244 C
4245          ENDIF
4246 C -------------------------------------------------------------------
4247          IF (IFlag_fail.eq.1) THEN
4248            Sigma_fail = Sigma_m(1,1)
4249          ELSEIF (IFlag_fail.eq.2) THEN
4250            Sigma_fail = Sigma_f(1,1)
4251          ENDIF
4252          WRITE(Phi_dat,'(/A,I6,A,I6,A,F6.1,A/A,F6.1,A)')
4253     +     'Constituent #',IFlag_failed,' failed in Cycle #',
4254     +     NOC_old+NOC,' at a stress of Sigma =',Sigma_fail,' MPa!',
4255     +     'The external applied stress is',Sigma_tot(1,1),' MPa!'
4256 C -------------------------------------------------------------------
4257          CLOSE(7)
4258          CLOSE(N_Input)
4259          CLOSE(N_Error)
4260          CLOSE(N_Load)
4261          CLOSE(Check_c(1))
4262          CLOSE(Check_c(2))
4263          CLOSE(Check_r)
4264          CLOSE(Check_i1)
4265          CLOSE(Check_i2)
4266          CLOSE(Phi_dat)
4267          CLOSE(N_Restart)
4268 C
4269          STOP
4270 C
4271        ENDIF
4272 C -------------------------------------------------------------------
4273 C
4274 C     --- End of DO Loop over the total number of cycles
4275 C
4276 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
4277 2000  CONTINUE
4278 C <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
C ------------------------------------------------------------------
C     Write results in array to file
C
      IF (IFlag_write.eq.0) THEN
C
        DO i=1, NOC-1
C
          write(Phi_dat,'(1x,I6,8(3x,d14.8))')
     +              NOC_old+i,(Results_damage(i,j),j=1,8)
C
        ENDDO
C
        DO i=1, NOC-1
C
          write(Check_r,'(1x,I6,2(3x,f12.6))')
     +              NOC_old+1,(EVF(i,j),j=1,2)
C
        ENDDO
C
      ENDIF
C ------------------------------------------------------------------
C
C ------------------------------------------------------------------------
C
C     Write all variables for RESTART facility
C
C ------------------------------------------------------------------------
C
C      IF (IFlag_Restart.eq.1) THEN
C
      NTC = Number_of_Cycles_old + Number_of_Cycles
      NTI = L_inc
      NOR = Number_of_Restarts
      NDP = Number_of_discretization_points
      WRITE(N_Restart,'(1x,I4)') 1000
      WRITE(N_Restart,'(2x,I24)') NOR
      WRITE(N_Restart,'(2x,I24)') NTC
      WRITE(N_Restart,'(2x,I24)') NTI
      WRITE(N_Restart,'(1x,I4)') 1100
      WRITE(N_Restart,'(2x,I24)')  NDP
      WRITE(N_Restart,'(3(3(2x,d24.16):/))')
     +                ((Sigma_mean(i,j),j=1,3),i=1,3)
C
C --- Write material properties
C
      WRITE(N_Restart,'(1x,I4)') 2000
      WRITE(N_Restart,'(2(2x,d24.16))') c_f,c_m
C
C     --- Matrix ---
C
      WRITE(N_Restart,'(1x,I4)') 2100
      WRITE(N_Restart,'(2x,9(2x,d12.6))') (Prop_m(i),i=1,9)
      WRITE(N_Restart,'(3(3(2x,d24.16)/))')
     +                                  ((Nu_m(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 2120
      WRITE(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
     +             ((((E_m_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
C
C     --- Fiber ---
C
      WRITE(N_Restart,'(1x,I4)') 2200
      WRITE(N_Restart,'(2x,9(2x,d12.6))') (Prop_f(i),i=1,9)
      WRITE(N_Restart,'(3(3(2x,d24.16):/))')
     +                                  ((Nu_f(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 2220
      WRITE(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
     +                 ((((E_f_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
C
C     --- Interface ---
C
      WRITE(N_Restart,'(1x,I4)') 2300
      WRITE(N_Restart,'(2x,9(2x,d12.6))') (Prop_i(i),i=1,9)
      WRITE(N_Restart,'(3(3(2x,d24.16):/))')
     +                                  ((Nu_i(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 2320
      WRITE(N_Restart,'(3(3(3(3(2x,d24.16):/))))')
     +                 ((((E_i_bar(i,j,k,l),l=1,3),k=1,3),j=1,3),i=1,3)
C
C --- Write damage parameters
C
C     --- Matrix ---
C
      WRITE(N_Restart,'(1x,I4)') 3000
      WRITE(N_Restart,'(2x,d24.16)') F_power_m
```

```
      WRITE(N_Restart,'(2x,d24.16)') c_Gamma_m
      WRITE(N_Restart,'(3(2x,d24.16))') (Lambda_m(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Eta_m(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Vi_m(i),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 3100
      WRITE(N_Restart,'(3(2x,d24.16))') (Xi_m(i),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') Kappa_m
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Phi_m(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') g_m
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Gamma_m(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Y_m(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') F_pot_m
C
C     --- Fiber ---
C
      WRITE(N_Restart,'(1x,I4)') 4000
      WRITE(N_Restart,'(2x,d24.16)') F_power_f
      WRITE(N_Restart,'(2x,d24.16)') c_Gamma_f
      WRITE(N_Restart,'(3(2x,d24.16))') (Lambda_f(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Eta_f(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Vi_f(i),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 4100
      WRITE(N_Restart,'(3(2x,d24.16))') (Xi_f(i),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') Kappa_f
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Phi_f(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') g_f
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Gamma_f(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Y_f(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') F_pot_f
C
C     --- Interface ---
C
      WRITE(N_Restart,'(1x,I4)') 5000
      WRITE(N_Restart,'(2x,d24.16)') F_power_i
      WRITE(N_Restart,'(2x,d24.16)') c_Gamma_i
      WRITE(N_Restart,'(3(2x,d24.16))') (Lambda_i(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Eta_i(i),i=1,3)
      WRITE(N_Restart,'(3(2x,d24.16))') (Vi_i(i),i=1,3)
      WRITE(N_Restart,'(1x,I4)') 5100
      WRITE(N_Restart,'(3(2x,d24.16))') (Xi_i(i),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') Kappa_i
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Phi_i(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') g_i
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Gamma_i(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2(3(2x,d24.16)/),3(2x,d24.16))')
     +                                ((Y_i(i,j),j=1,3),i=1,3)
      WRITE(N_Restart,'(2x,d24.16)') F_pot_i
C
C-------------------------------------------------------------------------------
C
      CLOSE(7)
      CLOSE(N_Input)
      CLOSE(N_Error)
      CLOSE(N_Load)
      CLOSE(Check_c(1))
      CLOSE(Check_c(2))
      CLOSE(Check_r)
      CLOSE(Check_i1)
      CLOSE(Check_i2)
      CLOSE(Phi_dat)
      CLOSE(N_Restart)
C
      END
C------------------------ End of MMF ------------------------------------
```

Vita

Rainer Echle was born in Gengenbach, Germany, on October 5, 1962. He received his diploma (Dipl.Ing.(FH)) from the Fachhochschule Karlsruhe (Germany), in January 1986. From January 1986 until December 1987 he was employed at the Fachhochschule Karlsruhe as a full-time assistant in the Department of Civil Engineering. Through a Fulbright Scholarship he was able to attend Graduate School at Louisiana State University from January 1988 until May 1990 when he received his Masters of Science degree in Civil Engineering. He joined the doctoral program at Louisiana State University in the Department of Civil and Environmental Engineering in January of 1991. He is presently a candidate for the degree of Doctor of Philosophy in Civil Engineering at Louisiana State University.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Rainer Echle

Major Field: Civil Engineering

Title of Dissertation: A Micro-Mechanical Fatigue Damage Model For Uni-Directional Metal Matrix Composites

Approved:

George Z. Voyiadjis
Major Professor and Chairman

John W. Leslie
Dean of the Graduate School

EXAMINING COMMITTEE:

J. A. Darrow

A. Reuland Kost

Su-Seng Pang

Jianhao Zhu

Date of Examination:

December 18, 1996